Technical Report
On
Concepts and Architecture of Context-Aware Collaborative Smart Objects for Ubiquitous Application

Version 0.4
July 16, 2007

By
Ubiquitous Computing Team,
Department of Computer Engineering,
Kyung Hee University,
Korea

In this technical report we propose mechanism for implementing context aware communication and collaboration services between everyday appliances to support ubiquitous application. Bridging the gap between the applications and hardware for smart every items we used distributed Tuple Space based Middleware.

"Ubiquitous computing names the third wave in computing, just now beginning. First were mainframes, each shared by lots of people. Now we are in the personal computing era, person and machine staring uneasily at each other across the desktop. Next comes ubiquitous computing, or the age of calm technology, when technology recedes into the background of our lives."

Mark Weiser

Abstract

Developments in wireless sensor networking have opened up new opportunities and pushing scenarios where application's intelligence is no longer relegated to the fringes of the system (i.e., on a data sink running on a powerful node) rather it is distributed within the WSN itself. The future will see the integration of everyday items with existing specialized technology (e.g. Medical) with pervasive, wireless networks. Specialized technology will collaborate with the installed infrastructure, augmenting data collection and real-time response. Examples of areas in which future medical systems can benefit the most from wireless sensor networks are in-home assistance, smart nursing homes, and clinical trial and research augmentation.

In this technical report we provide mechanism for implementing context aware communication and collaboration services between everyday items in order to provide ubiquitous application that possesses the essential elements of each of the applications. We used distributed Tuple space based Middleware to support Inter object collaboration and context aware communication also to facilitate the development of application.

This technical report is organized as follows, in *Chapter 1* we provide introductory background ubiquitous computing and our vision towards ubiquitous computing. In *Chapter 2* we provide concepts of Smart Object and how Smart Objects can collaborate in a smart environment. Our detailed architecture and introductory concepts of every module can be found on *Chapter 3*. In *Chapter 4* present our OS which is build on top of sensors followed by *Chapter 5,* where we discussed our Ubiquitous Networks which focuses on building energy-efficient sensor networks through the use of sensor platforms and processors in order to support out collaborative architecture. We have discussed our middleware on *Chapter 6.* In this chapter we also provide introductory concepts of Tuple Space. Inter object collaboration mechanism with the help of Middleware is also a part of this chapter. In Chapter 7 we discussed our distributed context repository which is based on Tuple Space. Chapter 8 deals with knowledge-processing mechanisms that can provide the same level of functionality as in the case of a centralized architecture. We also discussed our reasoning mechanism in this chapter. Our security mechanism is on Chapter 9 followed by conclusion on Chapter 10.

Smart collaborative everyday object networks may assist residents by providing memory enhancement, control of home appliances, medical data lookup, and emergency communication. We will present Ubiquitous Life Care as an application to show how this context aware collaborative architecture can be used to control everyday life.

Table of Contents

1. Introduction

1.1 Ubiquitous Computing Vision

The term "Ubiquitous Computing" was originally introduced by Mark Weiser [1] in the year 1991. In his fundamental article *"The Computer for the 21st Century"* [2], he elaborated about "the computer that disappears". For Weiser the way into the 21st century was obvious: Computer and Network technologies are getting smaller, cheaper, and more powerful. Therefore, more and more everyday artifacts are going to be equipped with a reasonable amount of computing power and, maybe even more important, are networked together into a virtually unique network of communicating "things that think". In the pure sense of the word, computing gets "ubiquitous", anywhere, any time. Computers in every thing that is calmly doing what we intend it to do, in a way that is non-obtrusive and user friendly, in a sense that we do not have to focus our attention on the trivia of running an electronic system.

Research on Ubiquitous Computing (Ubicomp) is related to many other disciplines from Robotics and Embedded Systems, Networking and Distributed Systems, to Artificial Intelligence and Psychology. Thus Ubiquitous computing is a very difficult integration of human factors, computer science, engineering, and social sciences.

1.2 Context-aware Computing

One goal of Context-aware Computing is to acquire and utilize information about the context of a device to provide services that are appropriate to the particular people, place, time, events, etc. For example, a cell phone will always vibrate and never beep in a concert, if the system can know the location of the cell phone and the concert schedule. Often, the term "Context-aware Computing" is used in a sense synonymously to Ubiquitous Computing. This is because almost every ubicomp application makes use of some kind of context. Ubicomp is mainly about building systems which are useful to users, which "...weave themselves into the fabric of everyday life until they are indistinguishable from it" [2].

For ubicomp systems, *Context* is essential. How can a system be helpful for a user? Users tend to move around often, doing new things, visiting new places, changing their mind suddenly, and changing their mood, too. Therefore, a helpful system seems to need some notion of *Context*.

*In the Human point of view***,** we have a quite intuitive understanding of *Context*. Here, *Context* is often referred to as "implicit situational understanding." In social interactions *Context* is of great importance. A gesture, a laugh, or the tone of sentences builds up the implicit "picture" of what is meant or what communication partner is thinking. The same words can have a completely different meaning in different contexts.

In Computer Science, Context is quite a familiar concept, be it within the discipline of Artificial Intelligence ("Thinking machines"), in Robotics ("Adaptive Systems"), in User Interface Design (like adaptive UIs or office assistants like the Microsoft Office assistant

called "Clippy"), or basically any other discipline (to some extent). Especially, every discipline dealing with human users tries to take into account human behavior one way or the other, with the generated output loops back as part of the vector of input values.
From the variety of definitions commonly used by Ubicomp researchers we can imagine how difficult it is to find a common ground. Context definitions are far away from mathematical precision and a particular definition often strongly depends on an authors' subjectiveness:

- Schilit and Theimer [3]: "Context is location, identities of nearby people and objects, and changes to those objects."
- A. Dey and Abowd [4]: *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."*
- Pascoe [5]: "Context is the subset of physical and conceptual states of interest to a particular entity."

So what is this leading to? Are those definitions helpful or misleading? In the sense of a functional definition they are only helpful as a general description of what to do. As an application designer they are only stating what they are doing anyway: trying to figure out what input is needed to produce the desired output. Hence, it is of topmost importance to have some common ground or a common "vocabulary" when talking about what *Context* is. We need some sort of *formal approach* towards handling and describing Context.

Furthermore, in a software engineering sense, we need building-blocks for building context-aware applications in a structured way. The Context Toolkit [6] by A. Dey is a step into this direction and a good example for this principle (fig.1.1.) The Toolkit includes building blocks called "Widgets", wrapper classes for Sensors which serve as a hardware-abstraction layer, "Aggregators", which concentrate multiple input values to a single output value, and "Interpreters", implementing some application logic and generating application dependant "higher-level" output based on the input given. They interpret the incoming data according to a pre-programmed scheme.

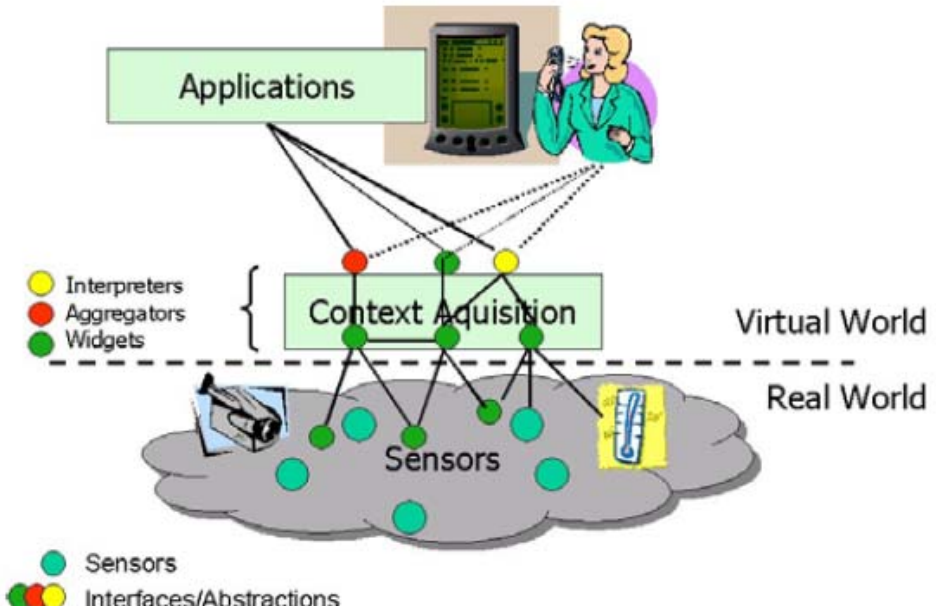


Figure 1.1: The Context Toolkit Core Components

With the Context Toolkit, the development of Context-aware applications basically consists of several distinguishable steps including

1) The real-world is sensed;
2) Context is detected, aggregated, "interpreted", and
3) Applications are custom-built to match the "context-detection" technology.

However, we believe that there is more tool-support necessary for software engineering and the design of Context-aware applications than provided today.

We want to emphasize that the way applications are developed is very dependant on the underlying technology used, which we consider as bad practice in the long run. Research in the direction of decoupling applications from data acquisition seems to be important.

1.3 Our vision

The continuous improvement of small devices capable of doing specific task and improvement of wireless communication have simulated the development and use of small wireless sensor network to perceive the physical environment. However, each single device may not be able to handle resource-intensive software with reasonable performance and quality. Therefore, we introduce an approach for this challenge which is to combine several devices to perform such tasks together or via personal computer and mobile agent technology.

That is, instead of running huge software like on a small device, one may connect several smart appliances together to form an ad hoc system. Also, we use mobile agent (e.g. handheld device) to join a distributed data structure (we call it distributed context repository) shared by cooperating smart devices, which makes the location where data are stored transparent for sensor applications. This allows smart appliance to transfer a graphical user interface or processed information to a nearby handheld, and facilitates the collaborative processing of sensory data because of the more elaborate storage and processing capabilities of mobile user devices. Fig. 1.2 shows an example application of collaborative processing of smart object and handheld.

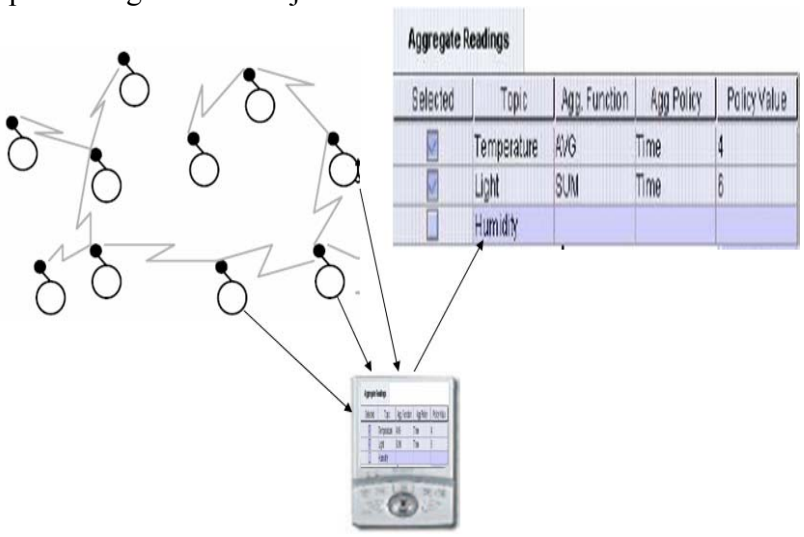


Figure: 1.2: An application example.

Typically, a network is formed by hundreds to thousands of sensor nodes, low power devices equipped with one or more sensors. Sensor nodes are responsible for collecting environmental information and sending it towards a sink node, which receives the information gathered by the network and delivers it to the final user.

These networks have been developed for a wide range of applications, such as habitat monitoring, object tracking, precision agriculture, building monitoring and military systems [2], [3], [4]. Common to these applications is the need of continuously collecting and integrating data coming from a large number of sensor nodes. Hence, a basic issue is

how to satisfy applications' requirements considering specific characteristics of sensor networks such as constrained sensor power and network bandwidth [5].

In this technical report our vision was to provide how sensor, smart sensors and smart everyday appliances can be used in collaborative approach for context aware for ubiquitous application. We have concentrated on:

- Collaboration between Desktop PC, Handheld devices, Smart object and smart sensors,

1.4 References

[1] Sungyoung Lee, Young Koo Lee, Anjum Shehzad, Hung N. Q., Kim Anh P. M. , Maria Riaz, Saad Liaquat. "Middleware Infrastructure For Context-Aware Ubiquitous Computing Systems"

[2] I.F.Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "A Survey on Sensor Networks", IEEE Communications Magazine, pp. 102-114, August 2002.

[3] A. Cerpa et al., "Habitat Monitoring: Application Driver for Wireless Communications Technology", ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean, Costa Rica, April 2001.

[4] G.J. Pottie, W.J. Kaiser, "Wireless Integrated Network Sensors", Communications of the ACM, Vol. 43, no. 5, pp. 51-58, May 2000.

[5] Eduardo Souto, Germano Guimarães, Glauco Vasconcelos, Mardoqueu Vieira, Nelson Rosa, Carlos Ferraz. "A Message-Oriented Middleware for Sensor Networks"

[6] Gelernter, David – "Generative communication in Linda" (published in ACM Transactions on Programming Languages and Systems, Vol 7 No 1, January 1985)

2. Context Aware Collaborative Smart Environment

2. 1 Smart Everyday Objects

A smart everyday object is an arbitrary item from our everyday environment – such as a chair, a hammer, a car, or an umbrella – augmented with information technology. Such augmentation must meet these requirements:

• **Unobtrusiveness**. The computation embedded into an object must not distract people from carrying out the task originally associated with it. The information technology should be either embedded in the design of an object or completely invisible – and not just a bulky add-on getting in the user's way.

• **Integrity**. People using a smart everyday object should perceive it as a single consistent unit. There should be a clear connection between the original purpose of the everyday object and the additional services provided by the embedded technology.

Information technology can either be directly integrated into the object itself, be available in a supporting background infrastructure, or both. If computation is provided by the environment rather than by the object itself, there must be a mechanism to link the object to a corresponding background infrastructure service that processes data and communicates with peers on behalf of the object.

The terms smart object and smart everyday object are used interchangeably throughout this thesis. However, considering definitions from the literature, there are significant differences between these two terms. Kintzig et al. [KPPF03], for example, defines a smart object as a "physical device equipped with a processor, memory, at least one network connection, and various sensors/actuators, some of which may correspond to human interfaces." According to their definition, electronic tags and wireless-enabled PDAs are all smart objects in their own right. This constitutes a major difference to our understanding of smart everyday objects: A smart everyday object always consists of both an everyday thing and information technology that augments it. PDAs and mobile phones themselves are therefore not smart everyday objects.

In the work presented in this thesis we concentrate on everyday things that are augmented by active sensor-based computing platforms. Such smart objects are able to perceive their environment through sensors, can carry out local computations, and can collaborate with other objects in their vicinity by means of wireless communication technologies. Furthermore, they possess an autonomous power supply and hence do not rely on external reader devices to supply energy.

Besides sensor-based computing platforms there are several alternative ways to link things with computation. Achieving this linkage usually requires some sort of electronic tag that is attached to ordinary objects. In the next section we present an overview of such tagging technologies, review how related projects augment everyday objects, and

compare their work to the tagging approach underlying our work [6]. Figure 2.1 shows a smart everyday object.

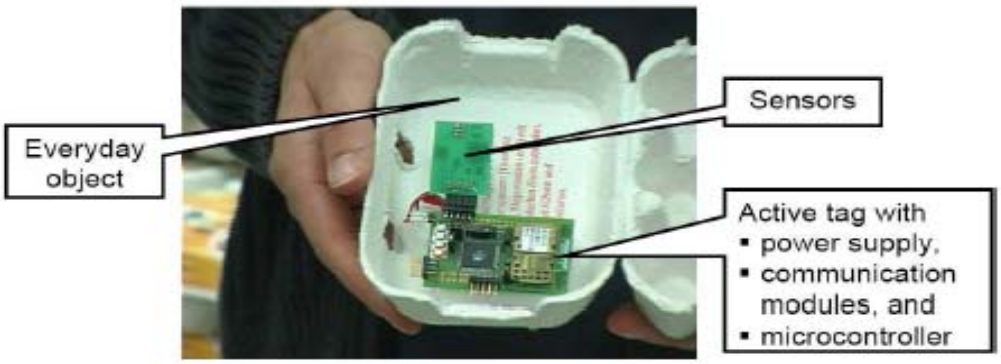


Figure 2.1: A smart everyday object: an everyday item augmented with a sensor-based computing platform.

2.2 Cooperation with other computing devices

Cooperation with other computing devices in smart environments helps smart objects to implement their services for several reasons. The most prominent one is context awareness. In smart environments, people expect applications to automatically adapt to their current situation, which requires smart objects to obtain information about their real-world environment and that of nearby users. The problem is that a single object can only perceive a small subset of its environment with its own local sensors, and therefore often needs to cooperate with other objects to more accurately derive context information. Another reason for cooperation is that the resources of smart objects are restricted as regards memory capacity and processing power. In order to carry out computationally expensive algorithms or to store large amounts of data, smart objects need to cooperate with other entities. Last but not least, the computing platforms attached to objects needs to be as unobtrusive as possible when augmenting everyday things with computation. In fact, prospective users should not even be aware of the integrated technology, and it must not hinder the way in which people normally use their items. Consequently, smart objects often have no sophisticated input/output interfaces such as displays or keys. This means that when a smart object wants to provide services that rely on such interfaces, it needs to cooperate with other computing devices – handhelds or mobile phones for example – that provide such user interfaces [1].

2.3 References

[1] Frank Siegemund. “Cooperating Smart Everyday Objects – Exploiting Heterogeneity and Pervasiveness in Smart Environments”

3. Our proposed architecture

Figure 3.1 shows our entire proposed architecture. It consists of five components which are virtually wired together. These are,

1. Sensor Hardware,
2. Sensor OS,
3. Ubiquitous Sensor Network,
4. Autonomic Context Aware middleware, and
5. Ubiquitous security.

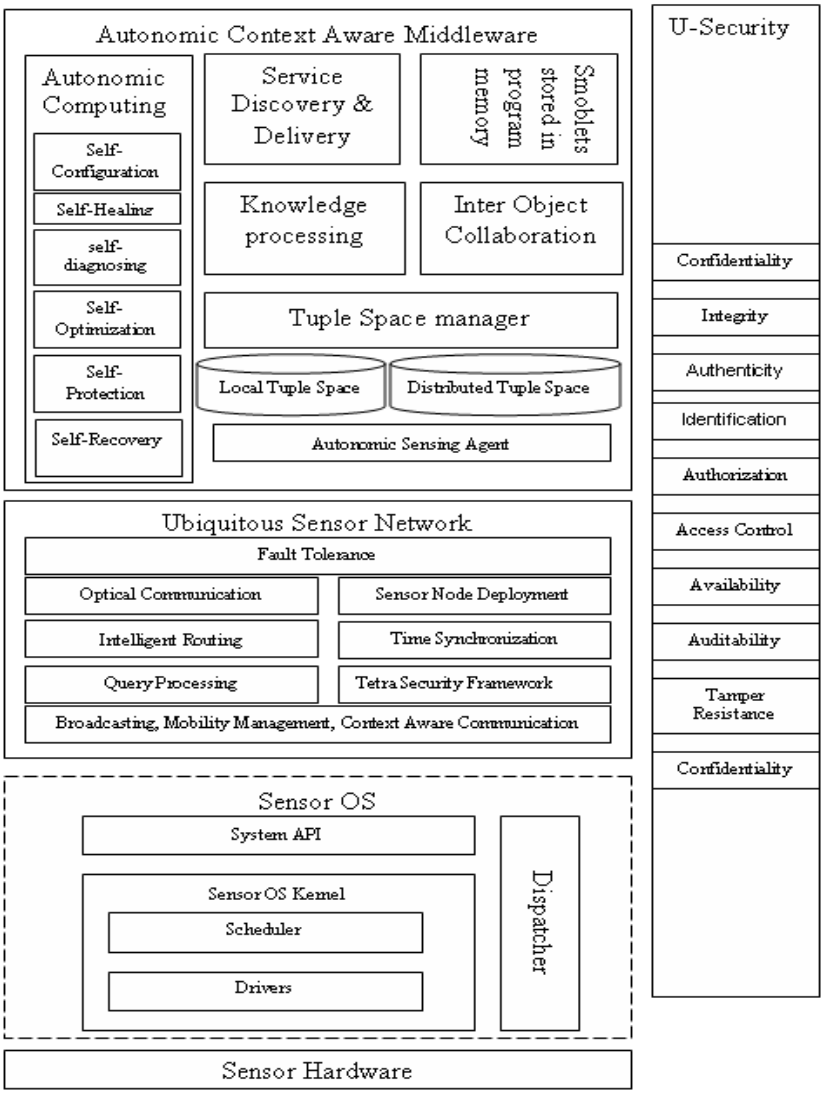


Figure 3.1: Our proposed architecture.

3.1 Sensor Hardware

We use ZigBee Sensors as our Sensor Hardware. The reasons using ZigBee sensors are,

- Standard in a fragmented market
  - Many proprietary solutions, interoperability issues
- Low Power consumption
  - Users expect battery to last months to years!
- Low Cost
- High density of nodes per network
- Simple protocol, global implementation

3.2 Sensor OS

Technological progress in integrated, low-power, CMOS communication devices and sensors makes a rich design space of networked sensors viable. They can be deeply embedded in the physical world and spread throughout our environment like smart dust. The missing elements are overall system architecture and a methodology for systematic advance. To this end, we identify key requirements, develop a small device that is representative of the class, design a tiny event-driven operating system, and show that it provides support for e_cient modularity and concurrency-intensive operation. Our operating system _ts in 178 bytes of memory, propagates events in the time it takes to copy 1.25 bytes of memory, context switches in the time it takes to copy 6 bytes of memory and supports two level scheduling. The analysis lays a ground work for future architectural advances.

In chapter 4, we describe details about our Sensor OS.

3.3 Ubiquitous Sensor Network

Ubiquitous Sensor network is one of the most challenging parts due to their unlimited potential. Ubiquitous Networks to support our collaborative context aware communication between Smart Objects and with handhaded.

Ubiquitous sensor networks (USN) as the underground infrastructure to achieve ubiquitous computing should be paid more attention.

The USN infrastructure provides a rich set of capabilities and exportable services to enhance and augment ubiquitous sensor networks applications.

In Chapter 5, we describe details about our Ubiquitous Sensor Network.

3.4 Autonomic Context Aware Middleware for Ubiquitous Computing System (aCAMUS)

Ubiquitous Sensor Network (USN) is evolving from centralized architecture to decentralized architectures where the application's intelligence is distributed among the smart everyday objects. Our proposed decentralized network consists of a large number

of sensors and actuators where the latter base their actions on the data gathered by the former. To support this decentralized network we proposed a distributed middleware architecture which is an autonomic by nature and supports applications where sensing and acting devices themselves drive the network behavior. In our proposed architecture application is not confined in a powerful node, rather it is deployed on the devices embedded within the physical environment.

In Chapter 6, we describe details about our aCAMUS architecture.

3.5 Ubiquitous Security

Generally, WSN devices are extremely limited in terms of power, computation, and communication. They are often deployed in accessible areas, thus increasing security vulnerabilities. The dynamic ad hoc topology, multicast transmission, location awareness, critical data prioritization, and co-ordination of diverse sensors of different applications further exacerbate the security challenges. We present an analysis of various WSN security mechanisms from the demanding perspective applications, and consider the importance of security in order to the successful accomplishment of Inter Object Collaboration.

We will be discussing more about this on chapter 9.

4. Sensor OS

4.1 Abstract

As we progress in time, computing devices are becoming essential part of human life. It all started from simple mathematical calculations. Initially, programs were written to assist human beings in carrying out simple and repetitive tasks. Intelligent tasks were carried out by human beings themselves. Then we moved into an era where the computing devices started taking up intelligent tasks also. From large computers, we moved onto personal computers, then Laptops. In parallel, humans invented machines, which can be used for communication e.g. Wireless phones. Wireless communication devices evolved to incorporate intelligence and smartness. In parallel to all this, sensor devices and their networks also paved their way into human needs. Combining all these things, we are now heading towards a new era, in which all these technologies will be merged together to form Smart Sensor Objects (SSOs). There is a need of having some distributed or networked operating system to make most efficient use of these devices.

4.2 Introduction

Ubiquitous computing aims at making computers available throughout the environment, while rendering them effectively invisible. One of the main goals is to incorporate computing power into everyday objects in order to create "smart things" real-world objects that provide novel ways of accessing information, react to their environment, or provide new emergent functionality when interacting with other smart things.

This vision of ubiquitous computing is grounded in the belief that microprocessors and advanced sensors will soon become so small and inexpensive that they can be embedded in almost everything. For ubiquitous environments, it is required that they should not render any change in the existing lifestyle of a user.

In SSOs, we have a certain hierarchy. At the lowest level, we have smart collaborative sensors (SCS), which are severely limited in resources. These sensors can be attached to crockery or even footwear. If we move a level higher, we find Smart Collaborative Objects (SCO). These devices are not as limited as SCS. The reason is that their energy resources can be replenished frequently.

However, SCOs are limited in other aspects like memory, computation and communication capabilities. The reason for these limitations has always been the cost associated with these devices. These devices are so large in number that these aspects must be taken into consideration. SCS are miniature devices, which sense and communicate with SCOs.

SCOs in turn communicate to higher level i.e. mobile devices like cellular phone. Cellular phones can be programmed to communicate with either a back-end server or some other device at some other place, to which it need to send information. The backend can be any computing device, with enough capabilities to store relative information and instruct the SCOs and SCS through mobile devices.

We organize our report as follows. In the next section, we will give an overview of SSO. In section 3, we will introduce some infrastructure concepts with respect to SSOs. In section 4, we discuss some common tasks required by SSO applications. This will be followed by a discussion on middleware concepts in SSOs in section 5. Kernel Reiability concepts will be discussed in section 6. Section 7 will cover existing system software techniques, which can be used in SSOs. We discuss incorporation of existing approaches into SSO paradigm in section 8. Section 9 states the conclusion and future plans of our team.

4.3 Overview

4.3.1. Smart Sensor Nodes

Smart objects are equipped with sensing, computation, and communication capabilities and are able to perceive and interact with their environment and with other smart objects.

These Smart Nodes participated in the European Smart-Its project whose goal was to develop unobtrusive, deeply interconnected smart devices that can be attached to everyday items in order to support new functionality, novel interaction patterns, and intelligent collaborative behavior. Smart-Its would be as cheap and as small as state-of-the-art radio tags (RFIDs), but in addition they will also be able to communicate with peers, and they will be customizable in their behavior. In order to facilitate a meaningful integration in their environment, Smart-Its are equipped with various sensors providing context information.

Figure 1: The Smart-Its project was one of 16 projects conducted under the European Union's Disappearing Computer initiative, whose aim was to explore how everyday life can be supported and enhanced through the use of collections of interacting artifacts.

Research groups at ETH Zurich and EPFL, are working on software architectures and infrastructures for mobile networks of smart, location-aware devices. The BTnode platform consists of a microprocessor, a Bluetooth radio, and various sensors. It is used as the basis of several prototypes built within the scope of this project.

Figure 2: The second generation of Smart-Its prototypes, called BTnodes, are based on an Atmel microcontroller with 128 kB of in-system programmable flash memory and 64 kB of SRAM. Bluetooth modules allow communication between different devices.

These groups are working on basic issues such as energy-efficient and scalable time synchronization and localization of sensor nodes, as well as on higher-level mechanisms for integrating these basic services into a common framework for the development of applications for sensor networks.

Many of their research topics are centered on this main theme: What kind of infrastructures is needed to support the interaction with and among smart objects? What key technologies can be used to prototype these visions with today's hard- and software? How to interact with smart objects, which might not have any perceivable user interface? How can hundreds of smart devices collaborate without knowing where they are or what exactly it is they do? And of course: How can security and dependability be provided at an adequate level in a world full of tiny interacting devices?

4.3.2 Infrastructures

To enable communication and cooperation among smart objects, new information infrastructures are required. These infrastructures have to cope with a highly dynamic environment and should, among other things, provide location information to mobile objects, represent context information, and enable reliable and scalable service creation.

Based on their experience with a collection of prototypical ubiquitous computing applications, they identified a number of common basic tasks which led to the design of some simple mechanisms that were found useful for structuring and implementing such applications. Building upon these mechanisms, they created a software infrastructure to support the application development of smart collaborating objects. Their framework was targeted at applications where objects were tagged with Radio Frequency Identification tags (RFIDs).

A central feature of their infrastructure is *virtual counterparts*. Virtual counterparts form augmented representations of real-world objects and encapsulate object states and object behavior in a computational entity. Virtual counterparts are a simple but powerful means of bridging the gap between the physical and the virtual world.

4.3.3 Technologies

Bluetooth: Bluetooth provides a standard wireless-communication interface and can carry data and voice over short ranges with relatively low power. In their research, they analyzed Bluetooth to suggest improvements and deduce requirements for emerging wireless-communication standards suitable for Ubiquitous Computing. The BTnode is a small, autonomous computing device combining computation, Bluetooth wireless communication and sensing. Augmentation of real-world objects with BTnodes, turn them into smart collaborating objects.

RFID: Their work on RFID was focused on the development of middleware concepts to support RFID systems rather than to design RFID systems themselves. They investigate how RFID can be used to bridge the gap between the physical and the virtual world.

4.3.4 Interaction

Interaction with the virtual aspects of an entity requires a device which is able to detect the entity and display related information. Such a "symbolic magnifying glass" allows, for example, to make object relations visible, to access virtual functions of an object, and to virtually zoom into and out of an object in order to get to know the object details or the object's context, respectively. Physical objects thus become "interactive" in a way which formerly was only possible with purely virtual, computer-based objects.

4.3.5 Context-Aware Collaboration

Handheld devices such as mobile phones or PDAs, computer-augmented everyday artefacts, RFID-enabled consumer products, and wall-sized displays are only some of the devices that are likely to play role in future smart environments. However, as pointed out by Mark Weiser, "the real power of the concept [of Ubiquitous Computing] comes not from any one of these devices; it emerges from the interaction of all of them." One core challenge in smart environments is therefore to exploit their heterogeneity by building applications that make use of and combine the specific capabilities provided by different types of computing devices.

Deriving the user's context from a set of collaborating smart objects distributed throughout the environment does therefore constitute a major part of a smart object's application behavior. Information about an object's own context makes it possible to dynamically form groups of collaborating artefacts, and to form networking structures that can make collaborative context recognition more efficient.

4.3.6 Dependability

However, with an ever growing number of smart devices and appliances, the probability of failure for any single device increases proportionally. They are investigating the role of redundancy and fault-tolerance in ubiquitous computing, with the goal to provide concepts for reliable infrastructures and services. Their main research interests in the field of dependable service infrastructures for ubiquitous computing include dependable location systems, reliability in highly dynamic smart object environments, and the roles of redundancy and connectivity in ubiquitous computing.

4.3.7 Security Concerns

Security and privacy will be of prime concern in a world of highly interconnected, invisible devices that will eventually permeate everyday life. They are working on a privacy-aware infrastructure (based on P3P, a W3C Web standard for exchanging privacy policies) that is able to keep track of any ongoing data exchange, while providing the user with powerful means to selectively manage such collected personal information. The operator of a wireless sensor network wants to make sure that outsiders cannot interfere with the operation of the network or gain access to confidential (or valuable) sensor data. Therefore, security mechanisms have to be implemented that protect the integrity, confidentiality, and availability of the network. Due to their potentially large size, existing security mechanisms are not always applicable to sensor networks. So they concentrate their research on localized protocols and methods for assessing security properties of sensor networks.

4.3.8 Applications of Smart Sensor Objects

- Smart Egg Carton
- Smart Medicine Cabinet
- Smart Product Packaging
- Smart Vacuum Cleaner
- Augmented Photoalbum
- RFID-Chef
- Smart Playing Cards
- Smart Surgical Kit
- Smart Toolbox

Website Reference: http://www.vs.inf.ethz.ch/res/show.html?what=sco can be visited for more details about the introduction of smart sensor objects.

Image Source: EU DC-Initiative

4.4 Infrastructure Concepts

The ultimate goal is to develop concepts and software frameworks to ease the development of ubiquitous computing applications. Researchers at ETH Zurich and EPFL have collected their experiences from the development of prototypical ubiquitous computing applications based on tagged physical objects. They have pointed out some tasks common to such applications. They have also mentioned some design concepts useful for structuring and implementing such applications. They have proposed an infrastructure based on these concepts.

4.4.1 Introduction

Objects brought into the vicinity of an antenna can be identified using tag reader devices working with tagging technologies like RFID. Some systems allow identification of multiple tags within one cubic meter. In the first prototype system built in this lab, only common thing was the RFID driver software. A number of tasks common to this type of application have been identified, which led to the design of concepts, useful for structuring and implementing applications using tagged physical objects..

4.4.2 Selected Applications

These are the applications that the researchers at ETH Zurich and EPFL have outlined and intend to support with their infrastructure. They have developed some of the RFID-based ubiquitous computing applications over the recent years. All the applications are based on multiple interacting tagged physical objects. These applications will serve as a basis for identifying common tasks that should be supported by a generic ubiquitous computing infrastructure.

RFID Chef: Grocery items are equipped with RFID tags. When placed on the kitchen counter, a nearby display suggests dishes that can be prepared with the grocery items available, or shows missing ingredients. The suggested dishes not only depend on the available ingredients, but also on the preferences of the cook, who might for example prefer vegetarian or Asian dishes. To implement this functionality, the cook is identified by an RFID tag attached to his wristwatch. When a person enters into the kitchen, possible dishes are suggested in the order of his preference.

Smart Playing Cards: Ordinary playing cards are equipped with RFID tags. An RFID antenna mounted beneath a table monitors the game moves of the players. A nearby display shows the score, the winner, and a cheat alarm if one of the players does not follow suit, and gives hints to beginners by assessing the players' moves. Each card must remember the contexts in which it was used and whether the trick in question was won or lost.

Smart Agenda: Agendas are equipped with RFID tags. If two or more people want to make an appointment, they place their agendas on the "appointment table", which is equipped with an RFID antenna. A nearby display shows possible dates that are compatible with the schedules of all the participants.

Smart Tool Box: Tools are equipped with RFID tags, and the tool box contains a mobile RFID system. The tool box issues a warning if a worker attempts to leave the building site (or a sensitive maintenance area such as an airplane) while any tools are missing from his box. The box also monitors how often and for how long tools have been in use. Based on this information, tools can be replaced before they wear out. Additionally, the tool owner can charge for tool rental based on actual tool usage.

Smart Medicine: This application helps to avoid trouble with medication by monitoring medicine from production to use. For this, medicine bottles are equipped with RFID tags. The environmental temperature of the medicine is constantly checked in order to avoid it going bad. Within the medicine cabinet, the bottle checks for other pharmaceuticals which are not compliant if taken together. A warning is issued if such dangerous situations are detected.

4.5 Common Tasks

4.5.1 Events

In order to enable an application to react to actions in the physical world, a link has to be established between tagged physical objects in the real world and the application. Since RFID systems detect presence and absence of tags in a certain physical space, this link can be established by notifying the applications about tags entering and leaving this space. A natural way to implement these notifications is by means of an event notification system. Also, applications need a way of expressing their interest in a subset of all possible tags, since a single antenna can be used by multiple applications at the same time.

4.5.2 Event Generation

Detection details for entering and leaving tags is a bit complicated. Periodic scanning technique and technique of replying after certain time slots is used in present systems. Not all scan results may be required. Only changes in the status of RFID tags are required. Moreover, false detection may occur due to collision or corruption at lower levels. For example, two replies of already present tags collide and both are not reported to the applications as being present while they are. Systems software should be used, which filter out such false detections.

Applications, such as smart playing cards, may require fast detection as the game is played in real time. Delayed detection may not be useful at all. Algorithms need to be designed and implemented, which are responsible for timely detection of entering/leaving tags.

4.5.3 Context

Application's action on entry/exit of a tag depends on the present context. For example, it may depend upon the presence or absence of other devices in the range. For instance in RFID chef application, when a new grocery item arrives, new list of dishes not only depend upon the current grocery items. It also depends upon the person, who has come to cook. Similarly, in smart playing cards, when a card is played, action also depends upon the cards already present on the table.

Sometimes, only a subset of current events is required in the context. For example in smart playing cards application, only last four or eight (in some cases) playing cards' presence will need to be known. This can be done at the application level, but it is better to push it as low as possible. This will increase performance and scalability of the system but it will come at some cost. May be, a proper expression of interest can make things lighter and practical in smart sensor objects such as sensor nodes.

4.5.4 Location

In RFID systems, the reading range of an RFID antenna is the location of a smart object. A tagged object is at this location if its tag can be read by the antenna. This idea is very simple and useful as it makes groups of tagged based objects based on their locations. In the Smart Tool Box application, for example, all the tools in the range of the tool box antenna belong to the same tool box. As in this example, "cooperating" physical objects are often located nearby to each other. For collocated objects to collaborate, they should adequately support the concept of neighbourhood. This might not be so easy because physically nearby objects may be on other sides of a wall and thus can't be each other's neighbours.

4.5.5 Composition

Some smart sensor objects might be a part of other smart objects. For instance medicine bottles can be smart objects and a medicine cabinet can also be a smart object. Medicine bottles are a part of the medicine cabinets. Some applications might be interested in the medicine cabinet as a whole i.e. it wants to perform certain operation on all objects inside the medicine cabinet. Object relationships can also be defined for such applications. Note that this is different from neighbourhood concepts as two neighbours might not be a part of same Smart Sensor Object.

4.5.6 Time

Time may also be important in many cases. A smart tool box might need to calculate the amount of time a tool was used so that its rent can be calculated. Smart playing cards may also need to know the temporal order in which the cards were played. Timestamps can be added by a common framework. There can be multiple tag readers. Framework should also support timestamps so that entry or exit recorded from any reader can be compared.

4.5.7 State and Behaviour

There can be applications, which support states of objects. These applications include smart tool box and smart agenda in which the states of tools and meetings are maintained. There can be applications that are stateless. In some applications, behaviour of smart objects is collective while in others, it is individual. For example in RFID chef, it is the behaviour of all objects collectively that reveal the list of available dishes. However, in case of applications like smart tool box, behaviour of individual tools also matter. In smart playing cards, individual behaviour of a card influences the context of all other cards on the table. This means that a flexible platform is required to handle the states and assign behaviour to physical objects.

4.5.8 History

Some applications may also need to know the history of entering/leaving of a smart sensor object. For example, tools might be queried about their usage history based upon their entrance/exit from tool box. This history can be maintained in a common infrastructure for all applications that require it.

4.5.9 Communication Infrastructure

Communication infrastructure might not be like the internet i.e. always connected. The application should be able to support offline operations. For example, tools might be connected to tool box through RFID infrastructure and the tool box can be offline. When it is taken back to the workshop and connected to a background communication infrastructure, it sends the collected information. This and any other sort of communication infrastructure should also be supported in the system software.

4.6 Middleware concepts

Following are some infrastructure concepts that support the above mentioned common tasks.

4.6.1 Virtual Counterparts

A virtual counterpart (VC) is a logical object that encapsulates the state and behaviour of a physical object such as a smart playing card. A virtual meta-counterpart is a logical object that collectively encapsulates the state and behaviour of a group of tagged physical objects such as grocery items. Some VCs are specialized to depict the range of an antenna. These are called Virtual Locations (VL). VLs are useful in measuring temperature and also Location management. There are also virtual meta-locations, representing a set of locations.

4.6.2 Counterpart events

Virtual Counterpart Event Services (VCES) is used to inform a virtual counterpart about an event of its physical counterpart. Physical counterparts produce events, virtual counterparts consume events. VCES forwards events from producers to consumers based upon their subscriptions i.e. events they are interested in. In the infrastructure software handling the physical events is the producer. More than one consumer can be interested in an event from a physical counterpart. Also, a consumer can be interested in the events of more than one physical counterpart. So m-to-n relationship needs to be managed between physical and virtual counterparts in the system.

Applications can also mention the context, in which they are interested in an event. The context will be implemented using a rule based mechanism in VCES. The event service will see both the event and the context, in which it is important to a consumer. If both match the rules, then the event is forwarded to the virtual counterpart. For example, a smart playing card application will need to see an event once a round in complete i.e. all four cards have been played. So other three events will not be forwarded in this case.

4.6.3 Counterpart Management

Counterpart management consists of Life-cycle management, location support and composition.

In Lifecycle management, a VC has to be instantiated when an object enters into the antenna range. It should be maintained i.e. its states should be managed with events that take place with the passage of time. The life-cycle management should also destroy the VC once the object leaves the vicinity for a long period of time. If the object comes up later, VC should also be re-instantiated from the saved state. Virtual locations are never destroyed after the first enter event from any of the participating RFID systems in the virtual location or virtual meta-location.

In location support, each VC is associated with its Virtual Location. Several physical locations can be grouped into a logical location using virtual meta-location mechanisms. VC can find out their neighbours using VLs.

Composition of VC is supported by arranging them in a hierarchy. Applications can use this hierarchy to perform operations on all objects contained in a composite object or to automatically update a composite object based on the changes in its contained objects.

4.6.4 Artefact Memory

Artefact Memory (AM) fulfills the task of maintaining the history of VCs. A very little number of states can be saved in tag memory. Larger histories are maintained in some database-like system. Using a database-like system, AM provides an interface for queries regarding tags and their locations at different times. AM can be useful to finding other counterparts and also depict behavioural patterns of certain tagged objects.

[1] Has detailed description about the infrastructure concepts of wireless sensor networks.

4.7 Kernel Reliability in Smart Sensor Objects

System software for SCOs should be able:

- Cope with highly dynamic environment.
- Provide location information to mobile objects.
- Represent context information.
- Enable reliable and scalable service creation.
- The developed framework should be targeted at applications where objects are tagged with Radio Frequency Identification tags (RFIDs)
- Device should be able to detect the entity, interact with it and display the related information.
- It should also be able to virtually zoom into and out of an object in order to get to know the object details or the object's context, respectively.
- Physical objects thus become "interactive" in a way which formerly was only possible with purely virtual, computer-based objects.
- Should be able to exploit the heterogeneity of the devices

- Should be able to combine the specific capabilities provided by different types of computing devices.
- Should be able to dynamically form groups of collaborating artifacts, and to form a network in a more efficient manner.
- As the number of smart devices grows, probability of failure of a single device increases proportionally, thus system software should be reliable, dependable and redundant to cope with the highly dynamic smart objects environments.
- Should provide privacy-aware infrastructure that is able to keep track of any ongoing data exchange, while providing the user with powerful aggregation features.
- Should implement Security mechanisms that protect the integrity, confidentiality, and availability of the network.
- Existing security mechanisms are not always applicable, so the research is aimed at localized protocols.

4.8 Existing Sensor Operating Systems

There may be many existing system software techniques that can be used in SSOs or tailored accordingly. However, we have found tinyOS as the most appropriate solution that can be used in this paradigm. Of course, we will need to do some tailoring before using it in SSO paradigm. We also think that it is necessary to mention about t-kernel. T-kernel provides us with some improvements over tinyOS's mate module. We can tailor and use t-kernel, where we have lesser limitations.

4.8.1 T-kernel

[2] has identified three features, which if incorporated in the wireless sensor networks (WSN), can improve the reliability. They will also facilitate the application development for the sensor nodes. In our opinion, these features can be of more interest to the kernel of Smart Sensor Objects (SSO) than wireless sensor networks. The reason is that energy constraints may not be as stringent in SSO as in WSN. Small computation overhead is the only concern in t-kernel as describes in [2].

The three features include OS protection, virtual memory and pre-emptive scheduling. In t-kernel, application and OS collaborate with each other to support the three features. With t-kernel, applications with typical workload in sensor networks, does not degrade much. T-kernel provides efficient binary translation, provides virtual memory without repeatedly writable swapping devices. It also protects from OS from application error without any memory protection or privileged execution hardware.

Load time modification approach is used to provide enhanced system abstraction. T-kernel has been implemented and studied on Berkeley MICA2 motes. T-kernel is found to be an order of magnitude faster than the virtual machine based approach.

Sensor Networks typically have 4 KB of RAM, which can not support needs for many applications. This emphasizes the use of virtual memory. With long running computation

tasks, periodic tasks will have to remain in queues longer. This will lead to timing discrepancies.

T-kernel is considered as not the only option for SSO. Since t-kernel is a flavor of TinyOS with some extra features, hence we are also motivated to study TinyOS keeping in mind the characteristics and limitations of smart collaboration environments. This will help us in developing a new flavor of TinyOS which can be considered as the best kernel for SSO.

4.8.1.1 Assumptions, challenges and approach of t-kernel

They have assumed that the hardware on which t-kernel will be deployed is reprogrammable. Large, non-volatile external storage and some RAM is available is available. It is recommended to have at least 4KB of RAM for efficient indexing and swapping.

T-kernel has a few challenges also. It has no hardware support and it is low in resources. Additionally, it also has potentially write-unfriendly external non-volatile storage. Keeping these constraints in mind, traditional design of protection, preemption and virtual memory is not viable. So, Lin Gu et. Al. has proposed a new design.

Load time processing is used to enhance the system abstraction. There are three reasons for this approach. First of all, we can't assume that a correct compiler has generated the code. Secondly, compilers can only perform static checks. Lastly, run-time techniques such as virtual machines have a lot of overhead.

4.8.1.2 Design of t-kernel

Application, in native instructions of the host node, resides in the external flash memory. T-kernel loads and runs from external flash and the application can be updated via wireless programming. T-kernel modifies the application code when loading the application. Each page of a program is modified i.e. *naturalized* before being dispatched. Modified code is called natin. Reason behind naturalization is that the code behaves properly and does not pose any threat to the operating system or the device. Application code is naturalized only once in its lifetime except if it is in a *bridging* process i.e. some branch destination address is rewritten.

Whenever a natin page is executed, it returns the control to the kernel, along with the information about next page. Within each natin page, cascading branch chain is used to speed up the search in jump instructions. In some cases, instead of transferring control to another natin via kernel, control is directly transferred. This process called bridging is done for speeding up transitions. Only 255 such transitions can be specified as 8-bit counter is used. This makes sure that the kernel does get hold of the CPU frequently.

Three level lookup is used for locating natins or kernel program counter. Virtual Program Counter (VPC) look-aside buffer is the fastest memory used. If there is a miss, a 2-associative VPC table is used following which is physical program memory as shown in Fig. 3. Whenever a kernel transition happens, it establishes a new stack on top of the

current stack for execution. Kernel heap is also isolated from program memory. Naturalization process ensures that the kernel area is protected.

Swapping is not very much encouraged in write-unfriendly memory systems. Normally, a flash area is useless after 10,000 writes. T-kernel provides flexibility to the developers to handle interrupts. It only catches the interrupt and services it using application provided handler. User is also provided a choice of using any instruction that is supported on the system directly.

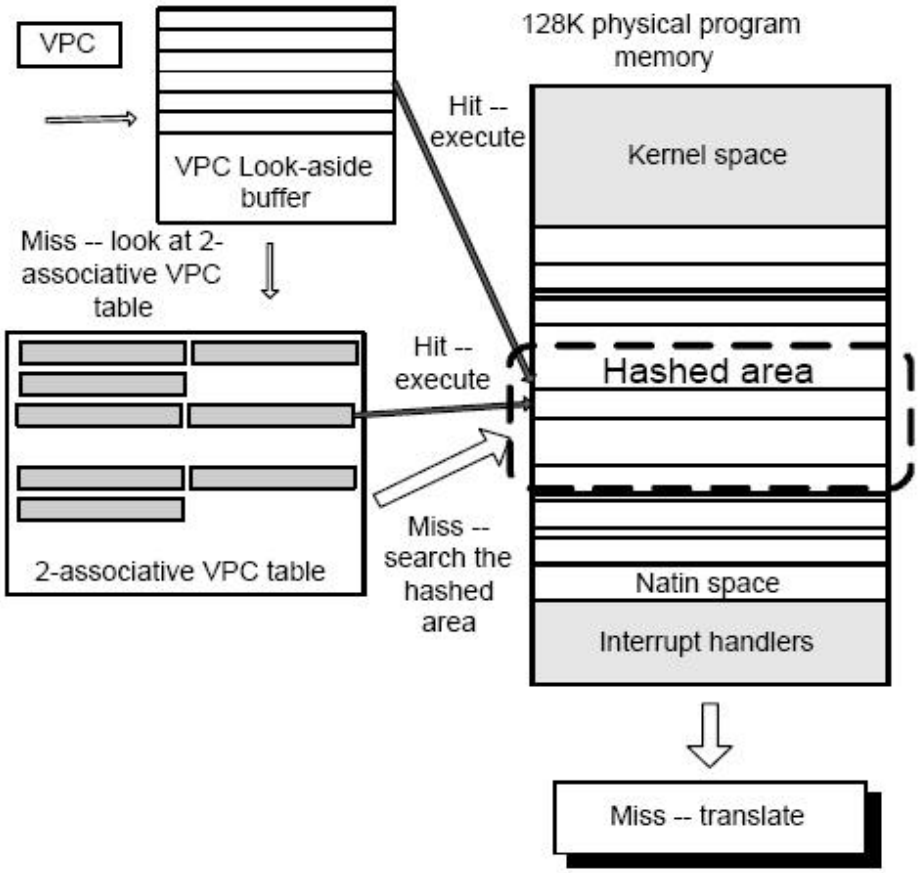


Figure 3: Three Level lookup scheme in t-kernel. Original figure can be found in [2].

4.8.2 TinyOS

TinyOS is a tiny (fewer than 400 bytes), flexible operating system built from a set of reusable components that are assembled into an application-specific system. TinyOS supports an event-driven concurrency model based on split-phase interfaces, asynchronous events, and deferred computation called tasks. TinyOS is implemented in the NesC language [24], which supports the TinyOS component and concurrency model as well as extensive cross-component optimizations and compile-time race detection. TinyOS has enabled both innovations in sensor network systems and a wide variety of applications. TinyOS has been under development for several years and is currently in its third generation involving several iterations of hardware, radio stacks, and programming

tools. Over one hundred groups worldwide use it, including several companies within their products.

The design of TinyOS, particularly its component model and execution model, addresses our four key requirements: limited resources, reactive concurrency, flexibility and low power. This section quantifies basic aspects of resource usage and performance, including storage usage, execution overhead, observed concurrency, and effectiveness of whole-system optimization.

4.8.2.1 Limited Resources

A TinyOS program's component graph defines which components it needs to work. Because components are resolved at compile time, compiling an application builds an application-specific version of TinyOS: the resulting image contains exactly the required OS services. TinyOS goes beyond standard techniques to reduce code size (e.g., stripping the symbol table). It uses whole-program compilation to prune dead code, and cross-component optimizations remove redundant operations and module-crossing overhead. To be efficient, TinyOS must minimize the overhead for module crossings. Since there are no virtual functions or address-space crossings, the basic boundary crossing is at most a regular procedure call.

4.8. 2. 2 Reactive Concurrency

TinyOS's component model makes it simple to express the complex concurrent actions in sensor network applications. The approach of allowing sophisticated handlers has proven sufficient for meeting these requirements; typically the handler performs the time-critical work and posts a task for any remaining work. With a very simple scheduler, allowing the handler to execute snippets of processing up the chain of components allows applications to schedule around a set of deadlines directly, rather than trying to coerce a priority scheme to produce the correct ordering.

4.8.2.3 Flexibility

TinyOS allows applications to be constructed from a large number of very fine-grained components. This approach is facilitated by cross module inlining, which avoids runtime overhead for component composition. Bidirectional interfaces and explicit support for events enable any component to generate events autonomously. In addition, the static race detection provided by NesC removes the need to worry about concurrency bugs during composition.

4.8.2.4 Low Power

The application-specific nature of TinyOS ensures that no unnecessary functions consume energy, which is the most precious resource on the node. The use of split-phase operations and an event-driven execution model reduces power usage by avoiding spinlocks and heavyweight concurrency (e.g., threads). The scheduler alone cannot achieve the power levels required for long-term applications; the application needs to convey its runtime requirements to the system. TinyOS address this requirement through a programming convention which allows subsystems to be put in a low power idle state. Components expose a StdControl interface, which includes commands for initializing,

starting, and stopping a component and the subcomponents it depends upon. Calling the stop command causes a component to attempt to minimize its power consumption, for example, by powering down hardware or disabling periodic tasks. The component saves its state in RAM or in nonvolatile memory for later resumption using the start command. It also informs the CPU about the change in the resources it uses; the system then uses this information to decide whether deep power saving modes should be used.

The component model of TinyOS can be exploited to build a new kernel flavor that can best suit the requirements of SSO and can help us transforming the vision of smart collaboration devices into a reality.

4.9 Discussion

We have discussed in the introduction that SSOs have a hierarchy that needs to be followed in order to make it usable in smart homes or smart offices. If we are unable devise a system software, that incorporate all devices from SCS to the mobile devices, then we might either put too much load on the lower level devices or we might not get best out of the higher level devices. We will have to make a lot of choices during the design phase.

One of such choices might be that either we design a distributed operating system or we choose to design a networked operating system. Former one will result in ease of application development and the later one will reduce the OS overhead, as it will not hide anything from higher level applications. We might have to use a mixture of both approaches due to the heterogeneity in SSO paradigm. In order to decide the design of the operating system for each level, we need to consider sensor access strategies, process scheduling algorithms and database management.

4.9.1 Sensor Access Strategies

There are many strategies, which can be applied to schedule the sensing issue in the nodes. These strategies clearly depend upon two things: capability of a node and requirements of the applications or middleware running on the node. Main sensor access strategies are briefly described as follows. They are described in much greater detail in the technical report of the aCamus team.

- **Best Effort:** In this case, the device accesses sensors as often as feasible. Most of the time, power considerations are taken into account in this strategy.
- **Fixed Effort:** In this case, time intervals are defined between which the sensor operating system polls for the input from sensors.
- **Random Interval:** In this case, the sensor operating system schedules the sensing based upon a random interval. Random interval is based upon a random number generated by the operating system.
- **Event Driven:** Sensor operating system schedules sensing events based upon certain events that occur in or around the sensor nodes. On one hand, this strategy may result in reduced power consumption. While on the other hand, it may also result in under-sampling i.e. not sufficient sampling to produce correct results.

4.9.1.1 Smart Collaborative Sensors

If we have to consider the power consumption only, we might consider the last strategy for lowest level smart collaborative sensors. However, this is not safe as it may result in under-sampling. Best effort strategy seems to be more viable for designing a sensor operating system kernel at this level. Perhaps, a mix of both these strategies can also be used. We can use best effort strategy in normal circumstances and event driven approach at the time when the smart collaborative sensors go below a certain energy level.

4.9.1.2 Smart Collaborative Objects

We can use better strategies at this level. Fixed effort sensor access strategy can also be introduced at this level if it improves sampling procedure. However, we can't go any further to introduce random interval strategy. This is because generating a random number requires considerable computation and storage overhead. Best effort and fixed effort strategies might compete for usage at this level. Decision might depend upon the energy limitations or computation, communication and storage capabilities of a device.

4.9.1.3 Mobile Devices

Sensor access strategies are not really applicable at this level. Mobile devices need not schedule sensor accesses. Rather, these devices will need to schedule communications with smart collaborative objects. There is a good chance that the original operating system of the mobile device has an optimized channel access or communication strategies.

4.9.2 Process scheduling Algorithms

Among the existing strategies, tinyOS has a very small image and simple process scheduling algorithm. Applications are compiled with this code to produce simpler images. Apart from that, we know about t-kernel that allows us to implement certain operating system features, which are found in conventional operating systems. Now we see application of such scheduling in each flavor of the sensor operating systems.

4.9.2.1 Smart Collaborative Sensors

Simplest process scheduling algorithms will be required at this level. We can't even think about adding a little bit of overhead. Original strategy of tinyOS might be the best to be used on this level, due to limited power resources.

4.9.2.2 Smart Collaborative Objects

As these devices have greater capabilities as compared to the smart collaborative sensors, we might choose to use some of the t-kernel features at this level. Protection and pre-emptive scheduling can be introduced in the design of sensor operating system at this level. All improvements might not be feasible, but can choose to have a go at some performance improvement strategies. These performance improvement strategies might depend upon any one factor, combination of more than one factor or even all the factors. The factors include storage, communication, computation and power resources.

4.9.2.3 Mobile Devices

We are relatively free to make design choices at this stage. This freedom is due to the fact that mobile devices are not as limited in resources as our SCOs or SCS. However, we are

not as free because we will have to use the existing devices. Existing devices have their own operating systems. Their existing operating systems might be good enough to relieve us from design considerations of scheduling algorithms at this level

4.9.3 Data management (tuple space)

Operating systems are designed to support higher layer applications or middleware. Likewise, sensor operating systems also have the same purpose. Data management is required at all levels. We might have to prune the required knowledge as we down in the hierarchy. This is mainly because of the limited capabilities of devices as we go down in the hierarchy. According to the knowledge processing team, this data management is based upon tuple spaces. Tuple spaces are better described in the knowledge processing part of this technical report. Management of data can be discussed in each flavor as follows.

4.9.3.1 Smart Collaborative Sensors

Smart Collaborative sensors are the lowest level devices. They have very limited power, storage, computation and communication capabilities. It is required that tuple spaces should be allocated to such devices in such a way that all these requirements are taken into account. Smart optimization strategies are required to carry out this complex task.

4.9.3.2 Smart Collaborative Objects

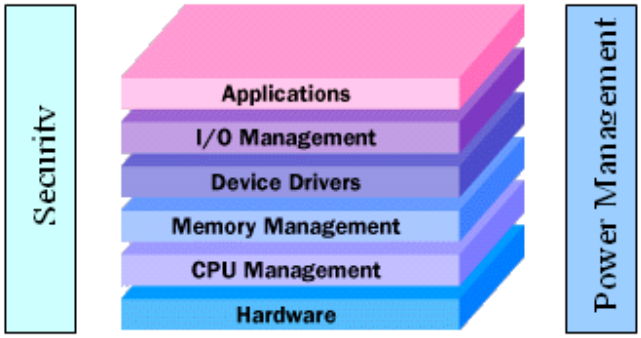
Smart Collaborative Objects are somewhat less limited in storage, computation, communication and energy capabilities. We can put more responsibility on such devices and relieve smart collaborative sensors from complexities. However, we should keep in mind that we smart collaborative sensors are not relieved in storage so much that communication takes away the lions share of available power.

4.9.3.3 Mobile Devices

System software design for mobile devices is not in our hands completely. We will need to take into consideration the original tasks that the device is designed to perform. However, we can give suggestions regarding the tailoring that can be done in the original operating system of the device. These changes will be completely dependent upon the capabilities of the devices working in such paradigm. At least, we can bring forward the requirements of our system from time to time, based upon the existing technologies used in the mobile devices.

4.10 Conclusion and Future Research

The simple operating system architecture, shown in the figure below, provides a research framework to support future research. A five year work plan is developed in this section specifically targeting SCO development. We have tried to identify the related tasks and then organized them into a five year plan.



Task 1: Conduct a detail survey of the necessary components of SCO sensor based networks including the technologies, infrastructures, roles and capabilities of these components. This survey should address all the layers shown in the above model with the integrated security and power management issues. It will also result in addition or subtraction of layers from the above initial model with respect to the requirements of SCO sensor based networks.

Task 2: Existing SCO applications will be studied in detail particularly their architecture, weaknesses and strengths. This survey will help us in proposing new applications with architectures lacking the previous weaknesses.

Task 3: Study an existing light weight sensor operating system in detail and analyze the feasibility of its application in SCO sensor based networks. The completion of this task will result in a deep understanding of TinyOS at different levels of our proposed design i.e. the outcome of the previous task. This survey will help us in designing different flavors of TinyOS for different levels of our architecture i.e. for Smart Cooperative Obejcts, Smart Cooperative Sensors and Mobile Devices.

Task 4: Developing the proposed application and the suggested new flavors of the sensor operating system.

4.11 References

[1] Kay Römer, Thomas Schoch. “Infrastructure Concepts for Tag-Based Ubiquitous Computing Applications”. Workshop on Concepts and Models for Ubiquitous Computing at Ubicomp 2002, Göteborg, Sweden, September 2002.
[2] L. Gu and J. A. Stankovic. “t-kernel: Providing reliable OS support wireless sensor networks”. In Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems (SenSys), 2006.
[3] Kay Römer, Thomas Schoch, Friedemann Mattern, Thomas Dübendorfer “Smart Identification frameworks for Ubiquitous Computing Applications”. Wireless Networks, Vol. 10, No. 6, pp. 689-700, December 2004.
[4] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An operating system for wireless sensor networks," in Ambient Intelligence. New York, NY: Springer-Verlag,

5. Ubiquitous Sensor Networks

5.1 Mobility-assisted Relocation for Self-deployment in Wireless Sensor Networks

5.1.1 Introduction

Sensor networks which are composed of tiny and resource constrained computing devices, have been widely employed for monitoring and controlling applications in physical environments [1]. Due to the unfamiliar nature of such environments, deployment of sensors has become a challenging problem and has received considerable attention recently.

Sensor deployment cannot be performed manually when the environment is unknown or inhospitable such as remote inaccessible areas, disaster fields and toxic urban regions. To scatter sensors by aircraft is one possible solution. However, using this scheme, the actual landing position cannot be predicted due to the existence of wind and obstacles such as trees and buildings. Consequently, the coverage may not be able to satisfy the application requirements. Some researchers suggest simply deploying large amount of static sensors to increase coverage; however it often ends up harming the performance of the network [5]. Moreover, there are situations where sensor deployment is restricted by the environment, for example, during in-building toxic-leaks detection [6] chemical sensors must be placed inside a building from the entrance of the building. In such cases it is necessary to take advantage of mobile sensors which can move to the appropriate places to provide the required coverage. This approach is different from the some of the work [2~4] which assume that the environment is sufficiently known and under control.

In this chapter, we introduce a comprehensive taxonomy framework for wireless sensor networks (WSN) self-deployment in which three sensor relocation algorithms are proposed according to the mobility degree of sensor nodes. The first one, particle swarm optimization based algorithm (PSOA), regards the sensors in the network as a swarm, and reorganizes the sensors by the particle swarm optimization (PSO) algorithm, in the full sensor mobility case. The other two, relay shift based algorithm (RSBA) and energy efficient fuzzy optimization algorithm (EFOA), assume relatively limited sensor mobility, i.e., the movement distance is bounded by a threshold, to further reduce energy consumption. Simulation results show that our approaches greatly improve the network coverage as well as energy efficiency compared with related works. The contributions of this chapter include the comprehensive collection of algorithms for mobile sensor network self-deployment within the context of a generally applicable taxonomy.

The rest of this chapter is organized as follows. Section 2 introduces related work and a comprehensive taxonomy framework integrating the different deployment schemes. Section 3 thoroughly explains the proposed three sensor relocation algorithms. In Section 4, extensive experiments and performance evaluations of the proposed method are presented. We conclude with a summary and discuss future work in Section 5.

5.1.2 Related Work and Taxonomy

In this section, we present a brief overview of the previous work on the coverage driven deployment of both stationary and mobile sensor networks that is most relevant to our study. A more thorough survey of the sensor network coverage is provided by [7].

We introduce a taxonomy framework (Fig 1) for WSN self-deployment in this section. We take the initial deployment as the first level, in which most of the existing research work makes an assumption of random distribution. Three categories of the full, limited and zero mobility are then considered as the top 2nd level of Fig. 1. The three relocation and scheduling cases for sensor network self-deployment corresponding to the three categories of sensor nodes mobility degree are extensively studied and integrated.

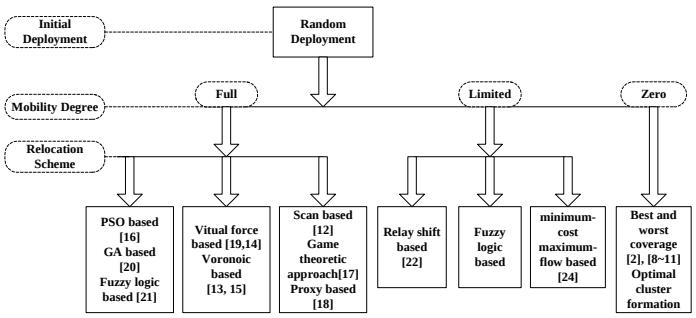


Fig. 1 Taxonomy framework integrating different deployment schemes.

In WSNs with zero mobility, i.e., stationary sensor networks, there are many previous studies which have focused on characterizing coverage. The authors of [8] consider a grid-based sensor network and derive the conditions for the sensing range and failure rate of sensors to guarantee that an area is fully covered. In [9], the authors propose several algorithms to find paths that are most or least likely to be detected by sensors in a sensor network. Path exposure of moving objects in sensor networks is formally defined and studied in [2], where the authors propose an algorithm to find minimum exposure paths, along which the probability of a moving object being detected is minimized. The best and worst coverage problem is explored in [10]. They propose an optimal polynomial time worst and average case algorithm for coverage calculation for homogeneous isotropic sensors. They also present several experimental results and analyze potential applications, such as using best and worst-case coverage information as heuristics to deploy sensors to improve coverage. In [11], the authors define several important coverage measures for a large-scale stationary sensor network, namely, the area coverage, detection coverage, and node coverage. Under the assumption that sensor location follows a Poisson point process, the authors obtain analytical results for the coverage measures under a Boolean sensing model and a general sensing model.

While the coverage of stationary sensor networks has been extensively studied and relatively well understood, a class of work has only recently appeared where full mobility of sensors is utilized to achieve desired deployment [12~21]. Typically in such works, the sensors detect lack of desired deployment objectives, then estimate new locations, and move to the resulting locations. In [19, 14], the authors propose a virtual-force-based

sensor movement strategy to enhance network coverage after an initial random placement of sensors. The virtual forces repel the nodes from each other and from obstacles to ensure that the initial configuration of nodes quickly spreads out to maximize coverage area. However they assume that global information regarding other nodes is available. Several distributed energy-efficient deployment algorithms are proposed in [15]. In order to achieve an energy-efficient node topology for a longer system lifetime, they employ a synergistic combination of cluster structuring and a peer-to-peer deployment scheme. Besides that, an energy-efficient deployment algorithm based on Voronoi diagrams is also proposed there. In [13], the authors propose several algorithms that identify existing coverage holes in the network and compute the desired target positions where sensors should move in order to increase the coverage. The main difference among all of their proposed algorithms is how the desired positions of sensors are computed. In [18], the authors propose a proxy-based sensor deployment protocol. Instead of moving iteratively, sensors calculate their desired positions based on a distributed algorithm, move logically, and exchange new logical locations with their new logical neighbors. Actual movement occurs at one time when sensors determine their final locations. The proposed protocol can greatly reduce the energy consumption while maintaining similar coverage. In [12], a scan-based movement-assisted sensor deployment method that uses scan and dimension exchange to achieve a balanced state is proposed. Using the concept of load balancing, it achieves good performance especially when applied to uneven distribution sensor networks. The authors of [17] study the dynamic aspects of the coverage of a mobile sensor network that depend on the sensor movement process. The results show that sensor mobility can be exploited to improve network coverage. For mobile targets, they take a game theoretic approach and derive optimal mobility strategies for sensors and targets from their own perspectives. In [20], the authors examine the optimization of wireless sensor network layouts using a multi-objective genetic algorithm (GA) in which two competing objectives are considered, total sensor coverage and the lifetime of the network. However the computation of this method is not inexpensive. In [21], fuzzy logic theory is applied to handle the uncertainty in full mobility sensor deployment problem. Their approach achieves fast and stable deployment and greatly increases the field coverage as well as communication quality. However, their fuzzy inference rules only consider two aspects, number of neighbors of each sensor and the average Euclidean distance between sensor node and its neighbors, without energy consumption included at all, which is one of the most critical issues in sensor networks.

In fact, the mobility of sensors is limited in most cases, as we have earlier discussed in [22]. To this extent, a class of Intelligent Mobile Land Mine Units (IMLM) [23] to be deployed in battlefields have been developed by Defense Advanced Research Projects Agency (DARPA). The IMLM units are employed to detect breaches, and move with limited mobility to repair them. This mobility system is based on a hopping mechanism and the hop distance is dependent on the amount of fuel and the propeller dynamics. Some other techniques can also provide such kind of mobility, for instance, sensors supplied by spring actuation etc. This type of model normally trades off mobility with energy consumption [24]. Moreover, in many applications, the latter goals outweigh the necessity for advanced mobility, making such mobility models quite practical in the future. In fact, [24] is one of the very few papers which deal with the mobility limited

deployment optimization. The mobility in the sensors they consider is restricted to a flip. However coverage is the only considered objective in their paper and their approach is not feasible in network partition case.

With the same goal as the existing research work in mind, that is, to improve the sensing coverage in a predefined area with low energy consumption and with connectivity guaranteed, we propose three different relocation algorithms, PSOA, RSBA and EFOA, in the cases of full sensor mobility and limited sensor mobility. We also indicate in the diagram that, in the zero mobility case, static topology control and scheduling schemes such as optimal number of cluster heads selection and cluster formation may be used. In general, cluster formation allows individual sensors to be grouped together for either communication or power efficiency. Cluster head is a node which manages the processing and relaying the information from its cluster members. In the next section, we will describe our proposed sensor relocation approaches in detail.

5.1.3 Proposed Relocation Schemes

We propose three different relocation methods for movement assisted self-deployment of sensors according to the mobility degree of sensor nodes. The common goal of the suggested schemes is to improve the sensing coverage in a predefined area with low energy consumption.

5.1.3.1 Relocation in Full Mobility Environment: PSOA

In the full sensor mobility case, we propose particle swarm optimization (PSO) based algorithm for movement assisted relocation. PSO, originally proposed by Eberhart and Kennedy [25] in 1995, and inspired by social behavior of bird flocking, has come to be widely used as a problem solving method in engineering and computer science [26~29].

All of particles have fitness values, evaluated by the fitness function to be optimized. PSO is initialized with a group of random solutions and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" factors. The first one, called *pbest*, is the best fitness it has achieved so far and it is also stored in memory. Another "best" value obtained so far by any particle in the population, is a global best and called *gbest*.

The PSO formulae define each particle in the D-dimensional space as $X_i = (x_{i1}, x_{i2}, x_{i3},\ldots\ldots,x_{iD})$ where i represents the particle number. The memory of the previous best position is represented as $P_i = (p_{i1}, p_{i2}, p_{i3}\ldots\ldots p_{iD})$, and the index of the best particle among all the particles in the population is represented by the symbol g. A velocity along each dimension is denoted as $V_i = (v_{i1}, v_{i2}, v_{i3}\ldots\ldots v_{iD})$. Let $d \in [1, 2, \ldots, D]$, the updating equation [30] is as follows**,**

$$v_{id} = \varpi \times v_{id} + c_1 \times rand() \times (p_{id} - x_{id}) + c_2 \times rand() \times (p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \quad (2)$$

where ω is the inertia weight, and c_1 and c_2 are acceleration coefficients.

The role of the inertia weight ω is considered to be crucial for the convergence of PSO. A suitable value for the inertia weight ω balances the global and local exploration ability and, consequently, reduces the number of iterations required to locate the optimum solution. Generally, it is better to initially set the inertia to a large value, in order to make better global exploration of the search space, and gradually decrease it to get more refined solutions. Thus, a time-decreasing inertia weight value is used.

PSO shares many similarities with genetic algorithm (GA). Both algorithms start with a group of a randomly generated population, have fitness values to evaluate the population with random techniques. Compared with GA, PSO is easier to implement, has fewer parameters to adjust, and requires only primitive mathematical operators. Because of its inexpensive computation and fast convergence rate, PSO is a potential algorithm to optimize deployment in a sensor network.

We assume that each node knows its position in the problem space, all sensor members in a cluster are homogeneous and cluster heads (CHs) are more powerful than sensor members. Sensing and communication coverage of each node are assumed to have a circular shape without any irregularity. The design variables are *2D* coordinates of the sensor nodes, $\{(x_1, y_1), (x_2, y_2), \ldots\}$. Sensor nodes are assumed to have certain mobility. PSOA includes two stages, the first is to optimize coverage by relocating sensors and the second is cluster formation when nodes have settled down during the first stage and don't move again.

5.3.1.1 Optimization of Coverage

We consider coverage as the first optimization objective. It is one of the measurement criteria of Quality of Service (QoS) of a sensor network.

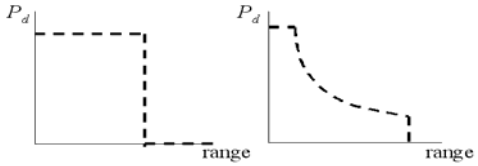


Fig. 2 Sensor coverage models (a) Binary sensor and (b) probabilistic sensor model.

The coverage of each sensor can be defined either by a binary sensor model or a probabilistic sensor model as shown in Fig. 2; both are used in this chapter. In the binary sensor model, the detection probability of the event of interest is 1 within the sensing range; otherwise, the probability is 0. In this case coverage is defined as the ratio of the union of areas covered by each node and the area of the entire Region Of Interest (ROI), as shown in Eq. (3) [15]. Generally ROI indicates the area in which sensor nodes need to be deployed. Here, the covered area of each node is defined as the circular area within its sensing radius. Perfect detection of all interesting events in the covered area is assumed.

$$C = \frac{\bigcup_{i=1,\ldots,N} A_i}{A} \tag{3}$$

where

A_i is the area covered by the i^{th} node;

N is the total number of nodes;
A stands for the area of the ROI.

In order to prevent recalculating the overlapped area, the coverage here is calculated using Monte Carlo method by creating a uniform grid in the ROI [16]. All the grid points located in the sensing area are labeled 1 otherwise 0, depending on whether the Euclidean distance between each grid point and the sensor node is longer or shorter than sensing radius, as shown in Fig 3. Then the coverage can be approximated by the ratio of the summation of ones to the total number of the grid points.

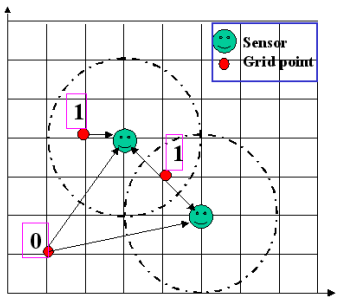


Fig. 3 Sensing coverage calculation (dashed circle indicating the sensing area boundary).

If a node is located well inside the ROI, its complete coverage area will lie within the ROI. In this case, the full area of that circle is included in the covered region. If a node is located near the boundary of the ROI, then only the part of the ROI covered by that node is included in the computation.

Although the binary sensor model is simpler, it is not realistic as it assumes that sensor readings have no associated uncertainty. In reality, sensor detections are imprecise, so that the coverage needs to be expressed in probabilistic terms. In many cases, cheap sensors such as omnidirectional acoustic sensors or ultrasonic sensors are used. Some practical examples [14] include AWAIRS at UCLA/RSC, Smart Dust at UC Berkeley, the USC-ISI network, the DARPA SensIT systems/networks, the ARL Advanced Sensor Program systems/networks, and the DARPA Emergent Surveillance Plexus (ESP). For omnidirectional sensors, a longer distance between the sensor and the target generally implies a greater loss in the signal strength or a lower signal-to-noise ratio. This suggests that we can build an abstract sensor model to express the uncertainty in sensor responses. In other words, a sensor node that is closer to a target is expected to have a higher detection probability about the target existence than the sensor node that is further away from the target.

$$c_{ij}(x, y) = \begin{cases} 0 & if \quad r + r_e \le d_{ij}(x, y); \\ e^{-\lambda a^{\beta}}, & if \quad r - r_e < d_{ij}(x, y) < r + r_e; \\ 1 & if \quad r - r_e \ge d_{ij}(x, y). \end{cases} \tag{4}$$

The sensor field is represented by a grid. An individual sensor node s on the sensor field is located at grid point (x, y). Each sensor node has a detection range of r. For any grid point P at (i, j), we denote the Euclidean distance between s at (x, y) and P at (i, j) as $d_{ij}(x,$

y), i.e., $d_{ij}(x, y)=\sqrt{(x-i)^2+(y-j)^2}$. Eq. (4) expresses the coverage $c_{ij}(x, y)$ of a grid point at *(i, j)* by sensor *s* at *(x, y)*, in which $r_e(r_e<r)$ is a measure of the uncertainty in sensor detection, $a = d_{ij}(x, y)-(r-r_e)$, and$\lambda$and$\beta$are parameters that measure detection probability when a target is at a distance greater than r_e but within a distance from the sensor. The distances are measured in units of grid points. In fact, the sensing behavior of almost all the omnidirectional range sensing devices including not only chemical sensors but also infrared, ultrasound, and acoustic sensors etc., can be modeled by probabilistic sensor detection model which is shown in Fig. 2(b). Fig. 2(b) also illustrates the translation of a distance response from a sensor to the confidence level as a probability value about this sensor response. The coverage for the entire grid sensor field is calculated as the fraction of grid points that exceeds the threshold c_{th}.

5.1.3.1.2 Optimization of Energy Consumption

After optimization of coverage, all the deployed sensor nodes move to their own positions. Our goal then becomes to minimize energy usage in a cluster based sensor network topology by finding the optimal cluster head (CH) positions. So cluster formation used to optimize energy consumption here is actually in a static sensor network manner. We are now in the second stage of PSOA.

According to the radio energy dissipation model, in order to achieve an acceptable Signal-to-Noise Ratio (SNR) in transmitting an *l* bit message over a distance *d,* the energy expended by the radio is given by [31]:

$$E_T(l,d)=\begin{cases} lE_{elec}+l\varepsilon_{fs}d^2 & if\ d\le d_0 \\ lE_{elec}+l\varepsilon_{mp}d^4 & if\ d>d_0 \end{cases} \tag{5}$$

where E_{elec} is the energy dissipated per bit to run the transmitter or the receiver circuit, ε_{fs} and ε_{mp} are amplifier constants, and *d* is the distance between the sender and the receiver. By equating the two expressions at $d=d_0$, we have $d_0=\sqrt{\varepsilon_{fs}/\varepsilon_{mp}}$. Here we set electronics energy as $E_{elec}=50nJ/bit$, whereas the amplifier constant is taken as $\varepsilon_{fs}=10pJ/bit/m^2$, $\varepsilon_{mp}=0.0013pJ/bit/m^2$.

In both cases, to receive l bit message, the radio expends:

$$E_R(l)=lE_{elec} \tag{6}$$

Assume that the sensor nodes inside a cluster have short distance *dis* to CH but each CH has long distance *Dis* to the base station. For each sensor node inside a cluster, to transmit an *l*-bit message a distance *dis* to CH, the radio expends

$$E_{TS}(l,dis)=lE_{elec}+l\varepsilon_{fs}dis^2 \tag{7}$$

For CH, however, to transmit an *l*-bit message a distance *Dis* to base station, the radio expends

$$E_{TH}(l,Dis)=lE_{elec}+l\varepsilon_{mp}Dis^4 \tag{8}$$

So the energy loss of a sensor member in a cluster is

$$E_s(l,dis)=l(100+0.01dis^2) \tag{9}$$

The energy loss of a CH is

$$E_{CH}(l, Dis) = l(100 + 1.3 \times 10^{-6} \times Dis^4) \quad (10)$$

Since the energy consumption for computation is much less than that for communication, we neglect computation energy consumption here.

Assume m clusters with n_j sensor members in the j^{th} cluster C_j. The total energy loss E_{total} is the summation of the energy used by all sensor members and all the m CHs:

$$E_{total} = l\sum_{j=1}^{m}\sum_{i=1}^{n_j}(100 + 0.01dis_{ij}^2 + \frac{100}{n_j} + \frac{1.3 \times 10^{-6} Dis_j^4}{n_j}) \quad (11)$$

Because only two terms are related to distance, we can just set the fitness function as:

$$f = \sum_{j=1}^{m}\sum_{i=1}^{n_j}(0.01dis_{ij}^2 + \frac{1.3 \times 10^{-6} Dis_j^4}{n_j}) \quad (12)$$

From Eq. (12) we can minimize the energy dissipation in the sensor network by reducing the distance from each node to its CH and the CH to the remote base station. We use the PSO algorithm to find the optimal CH positions in the sensor field when the minimized energy consumption is achieved.

5.1.3.2 Relocation in Limited Mobility Environment

5.1.3.2.1 Relay Shift Based Algorithm (RSBA)

Let *G(V, E)* be the graph defined on *V* with edges *uv* ∈E iff *uv* ≤*R*. Here *uv* is the Euclidean distance between nodes *u* and *v*, *R* is the communication range. We assume that sensor nodes know their locations using one of the GPS-less localization techniques mentioned in [13] such as received signal strength so that CH can get the position information of its sensor members.

We have 4 steps for implementing RSBA:

Step 1: Randomly deploy nodes in the network.

Step 2: Detect coverage holes and redundant sensor nodes. We set two distance threshold value T_1 and T_2. If the longest linear distance between two nodes A and B along the uncovered area perimeter is larger than T_1, regard it as a coverage hole, and create a virtual node point at the center of the straight line AB. If the distance between two neighbors is less than T_2, regard them as redundant nodes. Choose a redundant node nearest to the virtual node point in coverage hole.

Step 3: Use A* algorithm [32] to find a shortest path n_0-n_1-n_2-…-n_{n-1} from a redundant sensor n_0 to the destination n_{n-1} (added virtual node) in a coverage hole. The distance between n_{n-2} to n_{n-1} is bounded by *R*. A* algorithm is the most popular choice for pathfinding, because it is fairly flexible and can be used in a wide range of contexts. A* was developed to combine heuristic approaches like Best-First-Search (BFS) and formal approaches like Dijsktra's algorithm. It is like Dijkstra's algorithm in that it can guarantee a shortest path, while BFS cannot; and it is like BFS in that it works as fast as BFS which is faster than Dijsktra's algorithm. Take the advantage of A* algorithm, we can solve our problem more efficiently than our previous work [22] in which Dijsktra's algorithm was applied.

Step 4: Move sensor node n_{n-2} to the virtual node n_{n-1}, move n_{n-3} to n_{n-2} … finally move the redundant sensor n_0 to n_1, and leave the original location of sensor n_0 empty. The nodes coordinates can be updated by Eq (13):

$$NetLoc(n_i) = NetLoc(n_{i+1}), \quad i = 0, 1, \cdots, n-2 \tag{13}$$

$n_i \in$ nodes on shortest path from source to destination
n_0=source node
n_{n-1} =destination (virtual node)

The process is illustrated in Fig. 4 using an example of four sensors and one virtual node along the shortest path. Sensor node n_3 moves to the virtual node point n_4, n_2 moves to n_3 … finally the redundant sensor n_0 moves to n_1, and leave the original location of n_0 empty. The network coverage is defined and calculated the same using Eq (3).

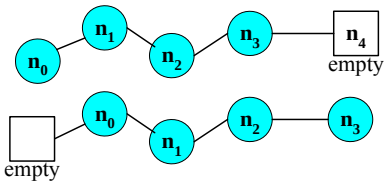


Fig 4. Illustration of sensor nodes relay shift along the shortest path

5.1.3.2.2 Energy-efficient Fuzzy Optimization Algorithm (EFOA)

A. Preliminaries of Fuzzy Logic System

The model of fuzzy logic system consists of a fuzzifier, fuzzy rules, fuzzy inference engine, and a defuzzifier. We have used the most commonly used fuzzy inference technique called Mamdani Method [33] due to its simplicity.

The process is performed in four steps:

1) Fuzzification of the input variables *energy, concentration and average distance to neighbors* - taking the crisp inputs from each of these and determining the degree to which these inputs belong to each of the appropriate fuzzy sets.

2) Rule evaluation - taking the fuzzified inputs, and applying them to the antecedents of the fuzzy rules. It is then applied to the consequent membership function.

3) Aggregation of the rule outputs - the process of unification of the outputs of all rules.

4) Defuzzification - the input for the defuzzification process is the aggregate output fuzzy set *moving distance* and the output is a single crisp number.

B. Energy-efficient Fuzzy Optimization Algorithm

The same energy dissipation model as Eq. (5) and Eq. (6) is used here for calculation of energy consumption. Assume an area over which *n* nodes are uniformly distributed. For simplicity, assume the sink is located in the center of the field, and that the distance of any node to the sink or its CH is $\leq d_0$ as explained in Section 3.1.2.

Two main procedures are carried out in our algorithm: 1) Determine the next-step move distance for each sensor. 2) Determine the next-step move direction for each sensor. Expert knowledge for deployment problem is represented based on the following three descriptors:

- Node Energy - energy level available in each node, denoted by the fuzzy variable *energy*,

- Node Concentration - number of neighbors in the vicinity, denoted by the fuzzy variable *concentration*,

- Average distance to neighbors - average Euclidean distance between sensor node and its neighbors, denoted by the fuzzy variable d_n.

The linguistic variables used to represent the node energy and node concentration, are divided into three levels: *low, medium* and *high*, respectively, and there are three levels to represent the average distance to neighbors: *close, moderate* and *far*, respectively. The outcome to represent the moving distance d_m is divided into five levels: *very close, close, moderate, far* and *very far*. The fuzzy rule base includes rules like the following: IF the energy is *high* and the concentration is *high* and the distance to neighbor is *close* THEN the moving distance of sensor node *i* is *very far*.

Thus we use $3^3 = 27$ rules for the fuzzy rule base. We use triangle membership functions to represent the fuzzy sets *medium* and *moderate* and trapezoid membership functions to represent *low, high, close, vclose, far,* and *vfar* fuzzy sets. The membership functions developed and their corresponding linguistic states are represented in Table 1 and Figures 5 through 8.

For the defuzzification, the Centroid is calculated and estimated over a sample of points on the aggregate output membership function, using the following formula:

$$Cen = \left(\sum \mu_A(x) * x\right) / \sum \mu_A(x) \tag{14}$$

where, $\mu_A(x)$ is the membership function of set *A*.

Table 1. Fuzzy rule base

No.	*En*	*Con*	d_n	d_m
1	low	low	close	close
2	low	low	moderate	vclose
3	low	low	far	vclose
4	low	med	close	moderate
5	low	med	moderate	close
6	low	med	far	vclose
7	low	high	close	moderate
8	low	high	moderate	close
9	low	high	far	close
10	med	low	close	moderate
11	med	low	moderate	close
12	med	low	far	close
13	med	med	close	far
14	med	med	moderate	moderate
15	med	med	far	close
16	med	high	close	far
17	med	high	moderate	moderate
18	med	high	far	moderate
19	high	low	close	far
20	high	low	moderate	moderate
21	high	low	far	moderate

22	high	med	close	vfar
23	high	med	moderate	far
24	high	med	far	moderate
25	high	high	close	vfar
26	high	high	moderate	far
27	high	high	far	far

Legend: vclose=very close, vfar=very far, med=medium, *En*=Energy, *Con*=Concentration

The control surface, or decision surface, is central in fuzzy logic systems and describes the dynamics of the controller and is generally a time-varying nonlinear surface. From Fig 9 and Fig 10 obtained by computation in Matlab Fuzzy Logic Toolbox [34], we can see that although the concentration for a certain sensor is high, the moving distance can be smaller than some sensor with higher energy or sensor with fewer neighbors but more crowded. With the assistance of control surface, the next-step moving distance can be determined.

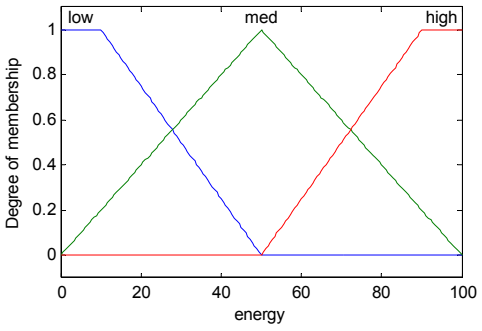


Fig. 5 Fuzzy set for fuzzy variable *energy*.

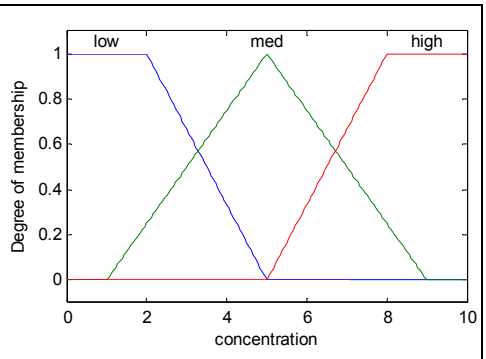
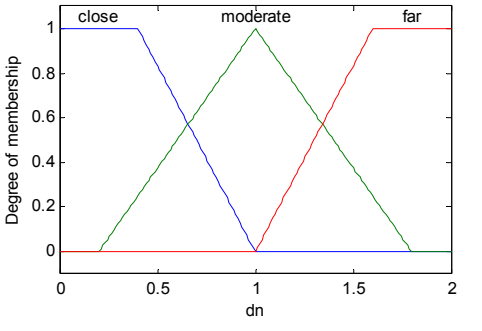


Fig. 6 Fuzzy set for fuzzy variable *concentration*.



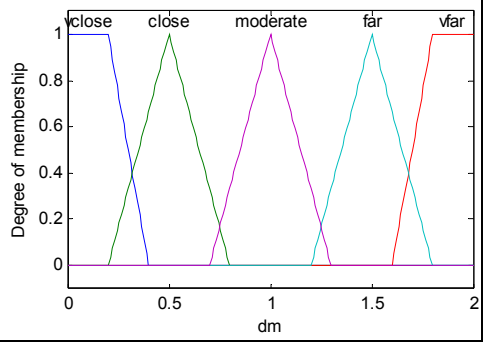


Fig. 7 Fuzzy set for fuzzy variable d_n.	Fig. 8 Fuzzy set for fuzzy variable d_m.

The next-step moving direction is decided by virtual force. Assume sensor i has k neighbors, $k=k_1+k_2$, in which k_1 neighbors are within threshold distance d_{th} to sensor i, while k_2 neighbors are farther than d_{th} distance to sensor i. The coordinate of sensor i is denoted as $C_i = (X_i, Y_i)$ and that of neighbor sensor j is $C_j = (X_j, Y_j)$. The next-step move direction of sensor i is represented as Eq. (15) and (16), thus sensor i clearly knows its next-step moving position by getting distance d_m and direction (angle α).

$$\vec{v} = \frac{1}{\left|\vec{C}_i - \vec{C}_j\right|^2}\left(\sum_{j=1}^{k_1}(\vec{C}_i - \vec{C}_j) + \sum_{j=1}^{k_2}(\vec{C}_j - \vec{C}_i)\right) \tag{15}$$

$$\tan(\alpha) = \frac{Y(\vec{v})}{X(\vec{v})} \tag{16}$$

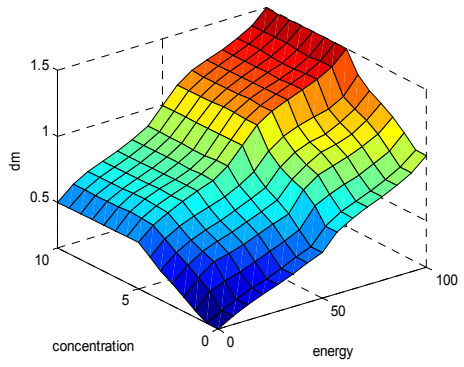


Fig. 9 Control surface (concentration, energy vs d_m)

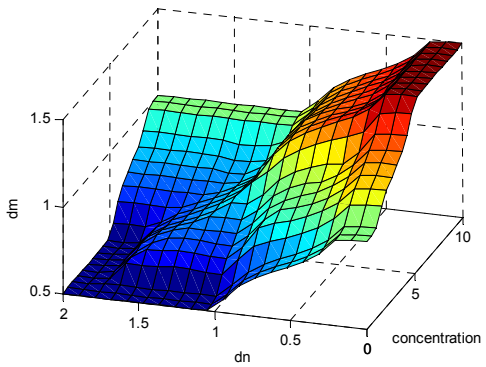


Fig. 10 Control surface (d_n, concentration vs d_m)

The threshold distance d_{th} here is set to a proper value $\sqrt{3}r$ which is proved as follows. We attempt to make distance between two sensor nodes moderate, i.e., not very close and not very far. This kind of stable structure is illustrated in Fig. 11. Non-overlapped sensor coverage style is shown in Fig. 11(a), however, an obvious drawback here is that a coverage hole exists which is not covered by any sensor. Note that an alternative way is to allow overlap, as shown in Fig. 11 (b) and it ensures that all grid points are covered. Therefore, we adopt the second strategy.

In Fig. 11(b), it is obvious that $\triangle S_1S_2S_3$ is equilateral triangle. Because the sensing radius is r, through some steps of simple geometry calculations, we can easily derive the distance between two sensor nodes in the latter case $S_1S_2 = S_2S_3 = S_1S_3 = 2\times\frac{\sqrt{3}}{2}r = \sqrt{3}r$.

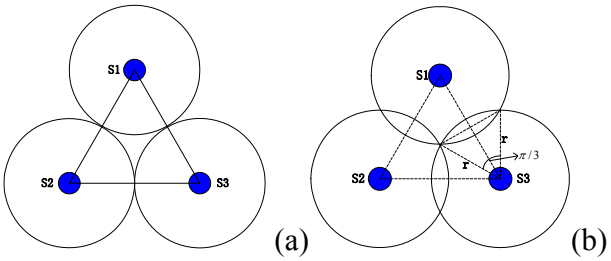


Fig. 11. Non-overlapped and overlapped sensor coverage cases

5.1.4 Performance Evaluations

5.1.4.1 Performance Evaluation of PSOA

5.1.4.1.1 Optimization of Coverage

A. Binary Model Case

The PSO starts with a "swarm" of sensors randomly generated. As shown in Fig. 12 is a randomly deployed sensor network with coverage value 0.4484 calculated using Eq. (3). A linear decreasing inertia weight value from 0.95 to 0.4 is used, decided according to [30]. Acceleration coefficients c_1 and c_2 both are set to 2 as proposed in [30]. For this performance study, we select a large scale deployment of 50×50 square sensor network. For optimizing coverage, we have used 20 particles, which are denoted by all sensor nodes coordinates and the maximum number of generations we are running is 500. The maximum velocity of the particle is set to be 50. The sensing range of each sensor is set to be 5 units. An upper bound on the coverage is given by the ratio of the sum of the circle areas (corresponding to sensors) to the total area of the sensor field. In this simulation, the upper bound evaluates to be 0.628, which is calculated from the perfect uniform distribution case without any overlapped area. The coverage is calculated as a fitness value in each generation.

Fig. 13 is the coverage optimization results. The coverage improvement verses number of iterations in one run is shown in Fig. 13 (a) and the final achieved coverage values for six runs are shown in Fig. 13 (b). Compared with the upper bound 0.628, the difference between the average value 0.58 for six runs and upper bound is small.

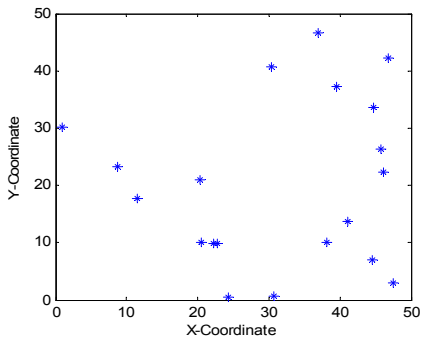


Fig. 12 Randomly deployed sensor network (Coverage value=0.4484).

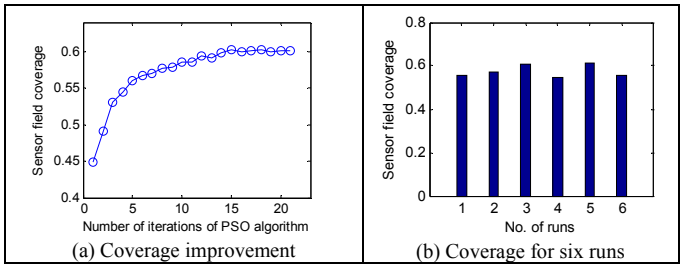


Fig. 13 Optimal coverage results (binary sensing model)

B. Probabilistic Model Case

In probabilistic model case, we use a randomly deployed sensor network as shown in Fig. 14, with coverage value 0.31 calculated by Eq. (4) and approximate method mentioned in section 3.1.1. PSO algorithm parameters are set the same as binary model case, however, the other parameters of sensor models are set to be r=5, r_e=3, λ=0.5, β=0.5, c_{th}=0.7.

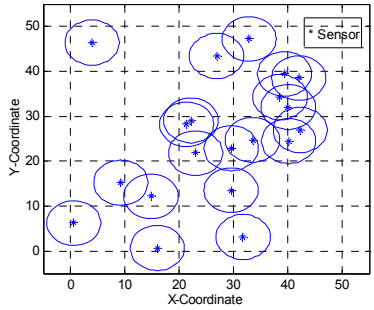


Fig. 14. Randomly deployed sensor network with r=5 (Coverage value=0.31)

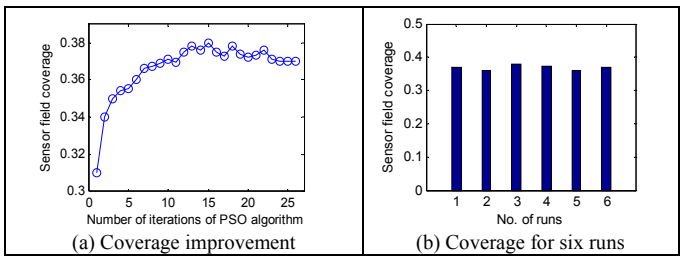


Fig. 15 Optimal coverage achieved using PSO algorithm (probabilistic sensing model)

Fig. 15 (a) shows the improvement of coverage during the execution of the PSO algorithm. Note that the upper bound for the coverage for the probabilistic sensor detection model (roughly 0.38) is lower than the upper bound for the case of binary sensor detection model (roughly 0.628). This is due to the fact that the coverage for the binary sensor detection model is the fraction of the sensor field covered by the circles. For the probabilistic sensor detection model, even though there are a large number of grid points that are covered, the overall number of grid points with coverage probability greater than the required level is fewer. We also show the achieved coverage for six runs

in Fig. 15 (b), and the average is nearly 0.37 which has little difference from the upper bound.

5.1.4.1.2 Optimization of Energy Consumption

After the optimization of coverage, all sensors move to their final locations in setup phase. Now the coordinates of potential CHs are set as particles in the sensor network. The communication range of each sensor node is 15 units with a fixed remote base station at (25, 80). We start with a minimum number of clusters acceptable in the problem space to be 4. The node, which will become a CH, will have no restriction on the transmission range. The nodes are organized into clusters by the base station. Each particle will have a fitness value, which will be evaluated by the fitness function (12) in each generation. Our purpose is to find the optimal location of CHs. Once the position of the CH is identified, if there is no node in that position then a potential CH nearest to the CH location will become a CH.

We also optimized the placement of CH in the *2-D* space using GA. We used a simple GA algorithm with single-point crossover and selection based on a roulette-wheel process. The coordinates of the CH are the chromosomes in the population. For our experiment we use 10 chromosomes in the population. The maximum number of generations allowed is 500. In each evolution we update the number of nodes included in the clusters. The criterion to find the best solution is that the total fitness value should be minimal.

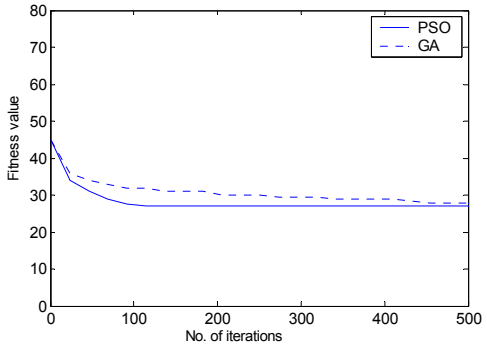


Fig. 16 Comparison of convergence rate between PSO and GA based on Eq. (12).

Fig. 16 shows the convergence rate of PSO and GA. We ran the algorithm for both approaches six times and in every run PSO converges faster than GA which was used in [20] for coverage and lifetime optimization. The main reason for the fast convergence of PSO is due to the velocity factor of the particle.

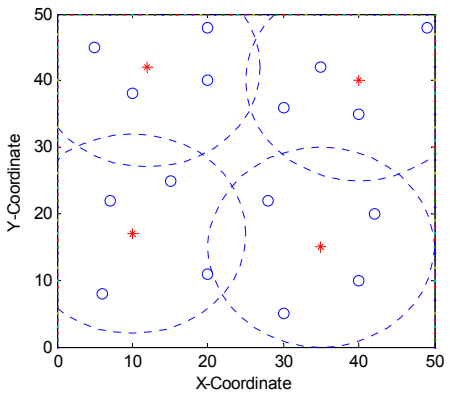


Fig. 17. Final cluster formation by PSO (Binary model case)

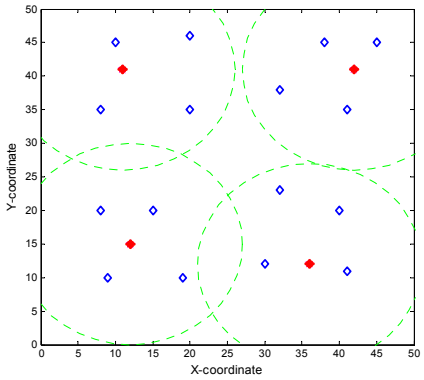


Fig. 18 Final cluster formation by PSO (Probabilistic model case).

Fig. 17 and Fig. 18 show the final cluster topology in the sensor network space after coverage and energy consumption optimization when the number of clusters in the sensor space is 4. We can see from the figure that nodes are uniformly distributed among the clusters compared with the random deployment as shown in Fig 12 and 13. The four red stars denote CHs, the blue tiny circles and diamonds are sensor members, and the dashed circles are communication range of sensor nodes. The energy saved is the difference between the initial fitness value and the final minimized fitness value. In this experiment, it is approximately 16.

5.1.4.2 Performance Evaluation of RSBA

The performance of the proposed movement assisted algorithm RSBA is evaluated by simulation. For the convenience of comparison with related work, we set the initial parameters the same as in [15]: 30 randomly placed nodes in a region of size 10×10 are used for initial deployment; the r and R used in the experiment are 2 and 4 m, respectively. In Fig. 19, the coverage and connectivity of the initial random deployment before running the algorithms are shown. The green circles are used to show the sensing range r of the nodes. Communications are possible within the R between nodes that are connected by a dashed line.

Fig. 20 shows the detected virtual node points (labeled as 31 and 32) in coverage hole and the redundant nodes nearest to 31 and 32 are 14 and 17 respectively. Both the coverage holes and the redundant nodes are judged by CHs. This information is then broadcasted by CHs to the whole network. The parameter values needed are: threshold value T_1=1.2 and T_2 = r/4.

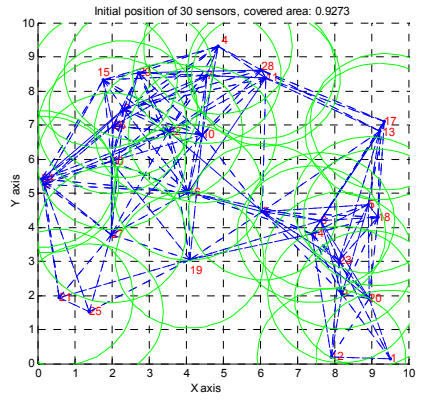


Fig. 19 Initial random deployment with sensing range 2m and communication range 4m.

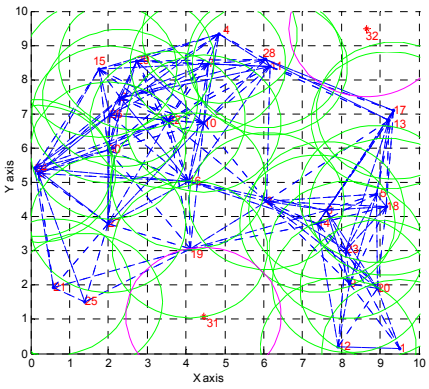


Fig 20 Determine virtual node point in uncovered area and redundant nodes.

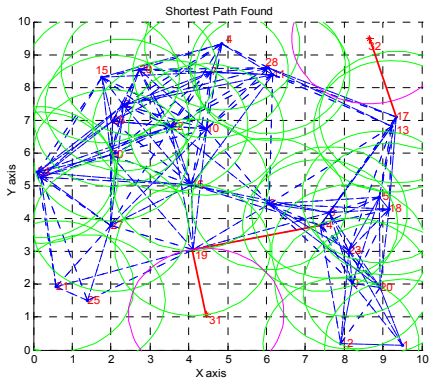


Fig. 21 Find shortest path by A* algorithm from redundant node to virtual node point.

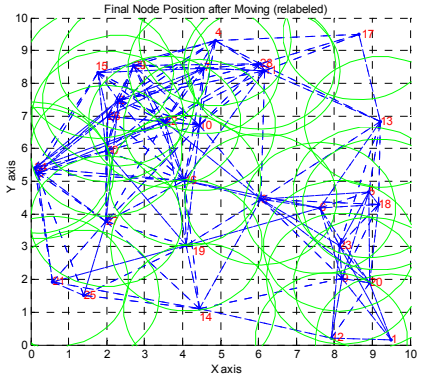


Fig. 22 Final node positions after executing proposed movement-assisted deployment algorithm

Fig. 21 shows the two shortest paths found (14→19→31 and 17→32) by A* algorithm from redundant nodes to virtual node points. This is also the actual path of individual nodes as they move by relay shift, in which sensor node move only one hop at a time which can guarantee the connectivity. For the initial distribution of Fig. 19, each node moves a distance of 2.6157 on average and the standard deviation of distance traveled is 0.5714. When the average distance traveled is small, the corresponding energy for locomotion is small. Also, when the standard deviation of distance traveled is small, the variation in energy remaining at each node is not significant and a longer system lifetime with desired coverage can be achieved. Fig. 22 shows the final node positions with desired coverage=0.9923 after executing RSBA. Note that the original 30 sensor nodes are finally reorganized and relabeled.

Next, the performances of RSBA are compared with DSSA, IDCA, and VDDA [15] in terms of coverage, movement distance until convergence, and time. Results are presented in Figures. 23–25. These results are obtained for different number of nodes dispersed over a fixed ROI of size 10×10, i.e., for different node densities to examine the relation between node densities and the performance metrics. The number of nodes varies from 20 to 40 and results are averaged over 10 runs (initial random distributions) for each node density.

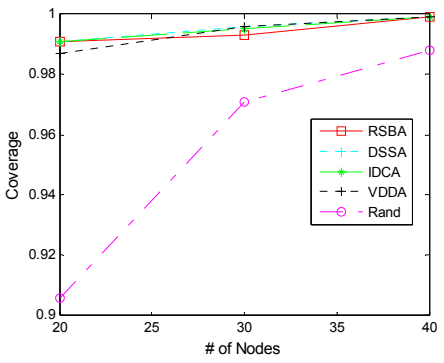


Fig. 23 Coverage comparison.

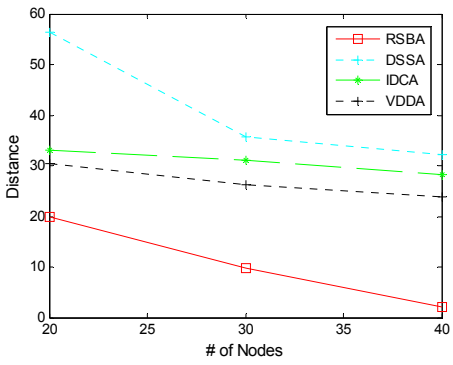


Fig. 24 Total distance traveled comparison.

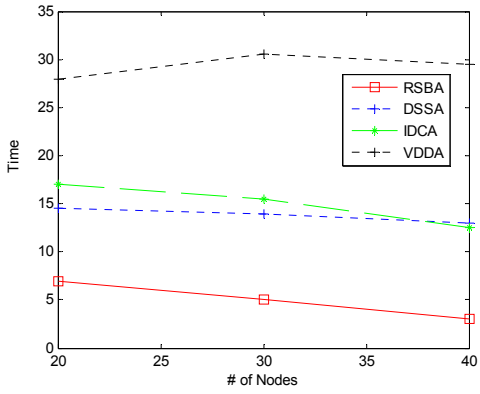


Fig. 25 Termination time comparison.

Fig. 23 shows the improvement in coverage area from the initial random deployment for RSBA, DSSA, IDCA, and VDDA. All four algorithms exhibit a similar performance. Although the coverage of RSBA (99%~1) is not always the highest among the four algorithms, this number is often satisfactory for many application requirements.

Fig. 24 shows the significant reduction of total distance traveled by RSBA compared with other three algorithms. In fact, distance moved here is used as the indicator of energy consumption. In RSBA, only very few numbers of nodes need to move and each sensor movement is bounded by only one hop. However, almost every node needs to move in the other three algorithms. So it is obvious that our proposed algorithm can save much

more energy compared with related methods. Fig. 25 shows that RSBA leads to faster deployment than DSSA, IDCA, and VDDA. Termination time is measured in the number of iterations until the algorithms stop.

5.1.4.3 Performance Evaluation of EFOA

For the convenience of comparison between EFOA and related work FOA [21], we set the initial parameters the same as in [21]: various number of sensors deployed in a field of 10×10 square area are investigated; the r and R used in the experiment are 1m and 2m (2m and 4m) respectively. So d_n should be ranged as 0~2 (0~4), not 0~10 as set by [21]. We assume each sensor is equipped with an omni antenna to carry out the task of detection and communication. Evaluation of our EFOA algorithm follows three criteria: field coverage, energy consumption and convergence. Results are averaged over 100 Monte Carlo simulations.

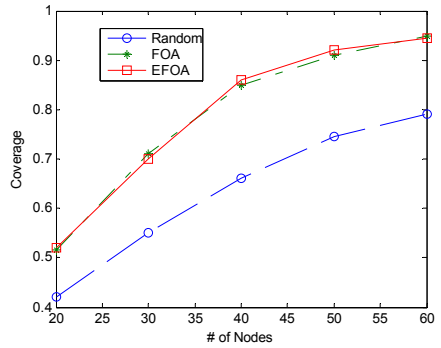


Fig. 26 Coverage vs. Number of Nodes (R=2, r=1).

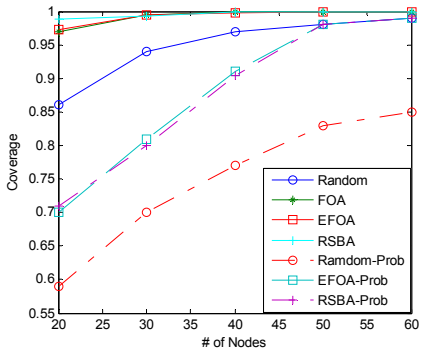


Fig. 27 Coverage vs. Number of Nodes (R=4, r=2).

Fig. 26 shows that the coverage of the initial random deployment, FOA and EFOA when r=1m and R=2m. The FOA and EFOA algorithms have similar results that both of them can improve the network coverage by 20% ~ 30% in average.

Fig. 27 gives the results when r=2m and R=4m, the coverage comparison of 1) random deployment, FOA, EFOA and RSBA with binary sensing model and 2) random deployment, EFOA and RSBA with probabilistic sensing model (denoted as Random-Prob, EFOA-Prob, and RSBA-Prob). In the case of binary sensing model, when 20 sensors are deployed, initially the coverage after random deployment is around 86%. After FOA and EFOA algorithm are executed, the coverage reaches 97%. RABA even has higher coverage ratio up to 99%. The coverage is dramatically improved in the low density network. The coverage ratio in case of probabilistic sensing model has similar improvement pattern by EFOA and RSBA compared with random deployment. The above two figures indicate that instead of deploying large amount of sensors, the desired field coverage could also be achieved with fewer sensors.

Fig 28 shows the total number of nodes that remain alive over time where each node begins with 2J of energy and when R=4 and r=2. The number of nodes in EFOA remains same for a long time and they die out quickly almost at the same time, while the first

node dies the earliest in FOA and RSBA in between. The reason is that after some operation time, the network display heterogeneous characteristics, however, FOA doesn't consider the residual energy of nodes, so the energy difference among sensors becomes significant as time goes on. Network lifetime is the time span from the deployment to the instant when the network is considered nonfunctional. When a network should be considered nonfunctional can be generally considered as the instant when the first sensor dies or a percentage of sensors die and the loss of coverage occurs. In RSBA, the uniformity is worse than EFOA but better than FOA. Thus the lifetime is prolonged in EFOA compared with FOA.

Fig. 29 shows EFOA has much lower standard deviation of distance compared with FOA in both cases when R=4, r=2 and R=2, r=1 while slightly higher than RSBA in the former case with various number of nodes. When the standard deviation of distance traveled is small, the variation in the energy remaining at each node is not significant and thus a longer system lifetime with desired coverage can be achieved. However, in case many sensors don't need to move in RSBA, although the standard deviation is low, it causes lower uniformity and thus slightly shorter lifetime compared with EFOA.

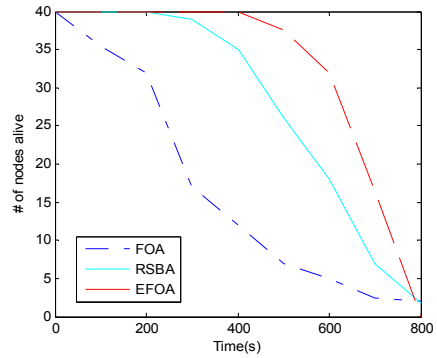


Fig. 28 # of nodes alive over time where each node begins with 2 J of energy. (R=4, r=2).

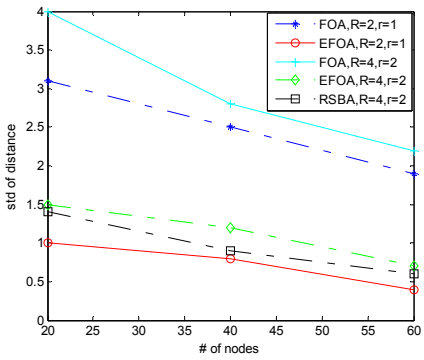


Fig. 29 Standard deviation of distance traveled verses number of nodes.

5.1.5 Conclusion and Future Work

In this chapter, we firstly introduced a comprehensive taxonomy for WSN self-deployment in which three sensor relocation algorithms were proposed according to the mobility degree of sensor nodes. The first one, PSOA, regards the sensors in the network as a swarm and reorganizes the sensors by PSO, in the full sensor mobility case. The other two, relay shift based algorithm (RSBA) and energy-efficient fuzzy optimization algorithm (EFOA), assume relatively limited sensor mobility, i.e., the movement distance is bounded by a threshold, to further reduce energy consumption. We also indicate in the diagram that in the zero mobility case static topology control and scheduling schemes can be used such as optimal cluster formation. Simulation results show that our approaches greatly improve the network coverage as well as energy efficiency compared with related works.

However in this chapter, only the discrete mobility metric cases were discussed. In the future work, we plan to study the abstraction of mobility degree in which both the physical factors (such as the friction of movement) and environmental factors (such as obstacles) will be included, so that the continuous mobility metric can be generated. Based on this metric we plan to design general deployment scheme. In addition, according to different situations, different types of initial deployment distribution such as Gaussian distributions can be further studied.

5.1.6 References

[] Xiaoling Wu, Hoon Heo, Riaz A. Shaikh, Jinsung Cho, Oksam Chae, and Sungyoung Lee, “Individual contour extraction for robust wide area target tracking in visual sensor networks”, Proc. 9th IEEE International Symposium on Object and component-oriented Real-time distributed Computing, Gyungju, Korea, pp. 179 – 185, Apr. 2006.

[2] S. Meguerdichian , F. Koushanfar, G. Qu and M. Potkonjak, “Exposure in wireless ad-hoc sensor networks”, Proc. of International Conference on Mobile Computing and Networking (Mobicom), Rome, Italy, pp. 139 – 150, 2001.

[3] S. Dhillon, K. Chakrabarty and S. Iyengar, “Sensor placement for grid coverage under imprecise detections”, Proc. International Conference on Information Fusion, Annapolis, USA, Vol. 2, pp. 1581 – 1587, Jul. 2002.

[4] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan and Kewal K. Saluja, “Sensor deployment strategy for detection of targets traversing a region”, ACM Mobile Networks and Applications, Vol. 8, Issue 4, pp. 453 – 461, Aug. 2003.

[5] Sameer Tilak, Nael B. AbuGhazaleh, and Wendi Heinzelman, “Infrastructure tradeoffs for sensor networks”, Proc. 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA), Atlanta, Georgia, USA, pp. 49 – 58, 2002.

[6] A. Howard, M. J. Mataric and G. S. Sukhatme, “An incremental self-deployment algorithm for mobile sensor networks”, Autonomous Robots, Special Issue on Intelligent Embedded Systems, Vol. 13, Issue 2, pp. 113 – 126, Sept. 2002.

[7] M. Cardei and J. Wu, “Coverage in wireless sensor networks”, in Handbook of Sensor Networks, M.Ilyas and I. Mahgoub (eds.), CRC Press, ISBN: 0-8493-1968-4, 2004.

[8] S. Shakkottai, R. Srikant, and N. Shroff, “Unreliable sensor grids: coverage, connectivity and diameter”, Proc. IEEE Conference on Computer Communications (INFOCOM), San Francisco, Vol. 2, pp. 1073 – 1083, Mar./Apr. 2003.

[9] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava, “Coverage problems in wireless ad-hoc sensor networks”, Proc. IEEE Conference on Computer Communications (INFOCOM), Anchorage, Alaska, USA, pp. 1380 – 1387, Apr. 2001.

[10] Seapahn Megerian, Farinaz Koushanfar, Miodrag Potkonjak, and Mani B. Srivastava, "Worst and best-case coverage in sensor networks", IEEE transactions on mobile computing, Vol. 4, No. 1, pp. 84 – 92, Jan. /Feb. 2005.

[11] B. Liu and D. Towsley, “A study on the coverage of large-scale sensor networks”, Proc. 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS’04), Fort Lauderdale, Florida, USA, pp. 475 – 483, Oct. 2004.

[12] J. Wu and S. Wang, “Smart: A scan-based movement-assisted deployment method in wireless sensor networks”, Proc. IEEE Conference on Computer Communications (INFOCOM), Miami, Vol. 4, pp. 2313 – 2324, Mar. 2005.

[13] G. Wang, G. Cao, and T. La Porta, “Movement-assisted sensor deployment”, IEEE Transactions on Mobile Computing, Vol. 5, Issue 6, pp. 640 – 652, June 2006.
[14] Y. Zou and K. Chakrabarty, “Sensor deployment and target localization in distributed sensor networks”, ACM Transactions on Embedded Computing Systems, Vol. 3, No. 1, pp. 61 – 91, Feb. 2004.
[15] Nojeong Heo and Pramod K. Varshney, “Energy-efficient deployment of intelligent mobile sensor networks”, IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems And Humans, Vol. 35, No. 1, pp. 78 – 92, 2005.
[16] Xiaoling Wu, Shu Lei, Yang Jie, Xu Hui, Jinsung Cho and Sungyoung Lee, “Swarm based sensor deployment optimization in ad hoc sensor networks”, Proc. 2nd International Conf. on Embedded Software and Systems (LNCS 3820), Xi’an, China, pp. 533 – 541, 2005.
[17] B. Liu, P. Brass, and O. Dousse, “Mobility improves coverage of sensor networks”, Proc. 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, Urbana-Champaign, Illinois, USA, pp. 300 – 308, May 2005.
[18] Guiling Wang, Guohong Cao, Tom La Porta, “Proxy-based sensor deployment for mobile sensor networks”, Proc. 1st IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS’04), Fort Lauderdale, Florida, USA, pp. 493 – 502, Oct. 2004.
[19] A. Howard, M. Mataric, and G. Sukhatme, “Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem”, Proc. of DARS'02, Fukuoka, Japan, pp. 299 – 308, June 2002.
[20] Damien B. Jourdan, Olivier L. de Weck, “Layout optimization for a wireless sensor network using a multi-objective genetic algorithm”, Proc. IEEE 59th Vehicular Technology Conference (VTC 2004-Spring), Milan, Italy, Vol.5, pp. 2466 – 2470, May 2004.
[21] Haining Shu, Qilian Liang, “Fuzzy optimization for distributed sensor deployment”, Proc. of IEEE Wireless Communications and Networking Conference, New Orleans, USA, pp. 1903 – 1907, 2005.
[22] Xiaoling Wu, Yu Niu, Lei Shu, Jinsung Cho, Young-Koo Lee, and Sungyoung Lee, “Relay shift based self-deployment for mobility limited sensor networks”, Proc. 3rd International Conf. on Ubiquitous Intelligence and Computing (LNCS 4159), Wuhan and Three Gorges, China, pp. 556 – 564, Sept. 2006.
[23] Defense Advanced Research Projects Agency (DARPA), “http://www.darpa.mil/ato/programs/shm/index.html”
[24] Sriram Chellappan, Xiaole Bai, Bin Ma, and Dong Xuan, “Sensor networks deployment using flip-based sensors”, Proc. of IEEE International Conference MASS, Washington, DC, pp. 291 – 298, Nov. 2005.
[25] Kennedy J. and R. C. Eberhart, “Particle swarm optimization”, Proc. IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942 – 1948, 1995.
[26] Ioannis N. Kassabalidis, Mohamed A. El-Sharkawi, Robert J. Marks, II, Luciano S. Moulin, and Alexandre P. Alves da Silva, “Dynamic security border identification using enhanced particle swarm optimization”, IEEE Transactions on Power Systems, Vol. 17, No. 3, pp. 723 – 729, Aug. 2002.
[27] Ganesh K. Venayagamoorthy and Sheetal Doctor, “Navigation of mobile sensors using PSO and embedded PSO in a fuzzy logic controller”, Proc. 39th IEEE IAS Annual Meeting on Industry Applications, Seattle, WA, USA, pp. 1200 – 1206, Oct. 2004.

[28] B. A. Kadrovach, G. B. Lamont, “A particle swarm model for swarm-based networked sensor systems”, Proc. ACM symposium on Applied computing, Madrid, Spain, pp. 918 – 924, 2002.
[29] T. Hardin, X. Cui, R. K. Ragade, J. H. Graham, and A. S. Elmaghraby, “A modified particle swarm algorithm for robotic mapping of hazardous environments”, Proc. the World Automation Congress, Spain, pp. 31 – 36, 2004.
[30] Yuhui Shi, Russell C. Eberhart, “Empirical study of particle swarm optimization”, Proc. the 1999 Congress on Evolutionary Computation, Washington DC, Vol. 3, pp. 1948 – 1950, 1999.
[31] Wendi B. Heinzelman, Anantha P. Chandrakasan, and Hari Balakrishnan, “An application-specific protocol architecture for wireless microsensor networks”, IEEE Transactions on Wireless Communications, Vol. 1, No. 4, pp. 660 – 670, 2002.
[32] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, “Introduction to algorithms”, 2nd Edition. MIT Press and McGraw-Hill, 2001.
[33] Indranil Gupta, Denis Riordan and Srinivas Sampalli, “Cluster-head election using fuzzy logic for wireless sensor networks”, Proc. 3rd Annual Communication Networks and Services Research Conf., Halifax, Canada, pp. 255 – 260, May 2005.
[34]The MathWorks, Inc., Fuzzy Logic Toolbox 2.2.3, “http://www.mathworks.com/products/fuzzylogic/”, 2006.

5.2 Localized Energy Aware Broadcast Protocol for Wireless Networks with Antennas

5.2.1 Introduction

In wireless networks which have limited resources such as sensor network, communication ranges are limited, thus many nodes must participate to the broadcast in order to have the whole network covered. The most important design criterion is energy and computation conservation, as nodes have limited resources. All the protocols that have been proposed for broadcast can be classified into two kinds of solutions: centralized and localized. Centralized solutions mean that each node should keep global network information and global topology. There exist several centralized energy-aware broadcast algorithms for the construction of broadcast trees with omni-directional antennas in the literature. In addition, the well-known energy-aware algorithm of Broadcast Incremental Power (BIP) [1] is "node-based" algorithm and exploits the "wireless broadcast advantage" property associated with omni-directional antennas, namely the capability for a node to reach several neighbors by using a transmission power level sufficient to reach the most distant one. Applying the incremental power philosophy to network with directional antennas, the Directional Broadcast Incremental Power (DBIP) algorithm [2] has very good performance in energy saving.

The problem of centralized approach is that any changes of nodes in their activity status (from active to passive and vice versa or dead) may cause global changes in topology which must be propagated throughout the network. This may results in extreme and un-acceptable communication overhead for networks. Hence, because of the limited resources of nodes, it is ideal that each node can decide on its own behavior based only on the information from nodes within a constant hop distance. Such distributed algorithms and protocols are called localized [3-7]. Of particular interest are protocols where nodes make decisions based solely on the knowledge of its 1-hop or 2-hops neighbors to them.

In this chapter, we propose and implement a localized energy-aware broadcast protocol which is based on the "Incremental Power" philosophy for wireless networks with Directional Antenna, Localized Directional Broadcast Incremental Power Protocol (LDBIP). Our localized protocol only uses localized and distributed location information and computing to construct broadcast tree. The use of directional antennas can reduce the beam width angle to diffuse the radio transmission to one direction and thus provides energy savings and interference reduction. In our algorithm, source node sets up spanning tree with only position information of its neighbors within two hops. Directional antennas are used for transmitting broadcast packet, and the transmission power is adjusted for each transmission to the minimal necessary for reaching the particular neighbor. Each node that receives broadcast packet will consider relay instructions included in received packet to compute its own two hops neighborhood spanning tree and do the same as source node. We compare the performance of our protocol (LDBIP) to those of BIP and DBIP. Experimental results show that in wireless networks, this new protocol has better performance compared to BIP, and similar performance to DBIP.

5.2.2 System Model

5.2.2.1 Network Model

We assume a wireless network consists of N nodes, which are randomly distributed over a specified region. Any node is permitted to initiate broadcast. Broadcast requests are generated randomly at network nodes. Each broadcast consists of the source node and at least one destination node. Additional nodes may be used as relays either to provide connectivity to all members in network or to reduce overall energy consumption. The set of nodes and the links of nodes support constructing a broadcast tree. Here, the links are incidental and their existence depends on the transmission power of each node. Thus, it is a set of nodes (rather than links) that are the fundamental units in constructing the tree. The connectivity of the network depends on the transmission power and antenna pattern. We assume that each node can choose its RF power level p^{RF} , such as $p_{\min} \le p^{RF} \le p_{\max}$. The nodes in broadcast tree can adjust their power levels for the various transmission in which it participates.

5.2.2.2 Propagation Model

We use two kinds of propagation model, free space model [8] and two-ray ground reflection model [9]. The free space model considers ideal propagation condition that there is only one clear line-of-sight path between the transmitter and receiver, while the two-ray ground model takes reality into consideration and considers both the direct path and a ground reflection path.

The following equation to calculate the received signal power in free space at distance *d* from the transmitter

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L}, \quad (2)$$

where P_t is the transmitted signal power. G_t and G_r are the antenna gains of the transmitter and the receiver respectively. L $(L \ge 1)$ is the system loss, and λ is the wavelength.

The following equation to calculate the received signal power in Two-ray ground model at distance *d*

$$P_r(d) = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4 L}, \quad (3)$$

where h_t and h_r are the heights of transmit and receive antennas respectively.

However, the two-ray model does not give a good result for a short distance due to the oscillation caused by the constructive and destructive combination of the two rays, whereas, the free space model is still used when *d* is small.

Therefore, a cross-over distance d_c is calculated. When $d < d_c$, Eqn. (1) is used. When $d > d_c$, Eqn.(2) is used. At the cross-over distance, Eqns. (1) and (2) give the same result. So d_c can be calculated as

$$(4\pi h_t h_r)/\lambda \quad . \tag{4}$$

When considering omni-directional antennas and uniform propagation conditions, it is common to select G_t and G_r as 1.

The use of directional antennas can permit energy savings and reduce interference by concentrating transmission energy where it is needed. We learn from [10] that because the amount of RF energy remains the same, but is distributed over less area, the apparent signal strength is higher. This apparent increase in signal strength is the antenna gain. We use an idealized model in which we assume that all of the transmitted energy is concentrated uniformly in a beam of width θ, as shown in Fig. 1, then the gain of area covered by the beam can be calculated as

$$2/(1-\cos\frac{\theta}{360}) \quad , \tag{5}$$

while the gain of the other areas is zero.

As a consequence of the "wireless broadcast advantage" property of omni-directional systems [11], all nodes whose distance from Node *i* does not exceed r_{ij} will be able to receive the transmission with no further energy expenditure at Node *i*.

While using directional antenna, the advantage property will be diminished, since only the nodes located within the transmitting node's antenna beam can receive the signal. In Fig. 1, only *j*, *l* can receive the signal, while *k* cannot receive the signal.

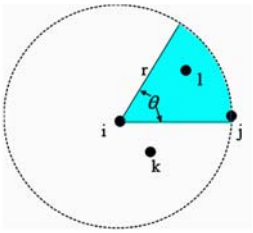


Figure 2.1: Use of Directional Antenna

Based on this model, the transmitted power required to support a link between Node *i* and Node *j* is

$$p_{ij}^{RF} = \text{RF power needed for link between Nodes } i \text{ and } j \tag{5}$$

$$= \max\{P_r(d,\theta), p_{\min}\}$$

The use of a nonzero value of $p_{\min}$ is that at the physical layer of each wireless sensor node, there is a receiving threshold. When a packet is received, if its signal power is below the receiving threshold, it is marked as error and dropped by the MAC layer.

We assume that the beam width θ is fixed beam width and one node can simultaneously support more than one directional antenna. Furthermore, we assume that each antenna beam can be pointed in any desired direction to provide connectivity to a subset of nodes that are within communication range. In addition, we use directional receiving antennas, which have a beneficial impact to avoid background noise and other user interferences.

5.2.2.3 Energy Expenditure

In addition to RF propagation, energy is also expended for transmission (encoding, modulation, etc.) and reception (demodulation, decoding, etc.). We define:

- p^T = transmission processing power and
- p^R = reception processing power.

The total power expenditure of a node, when transmitting to a maximum range r over a sector of width θ, is

$$p = p^{RF}(r,\theta) + p^T + p^R \quad (6)$$

Where $p^{RF}(r,\theta)$ is RF propagation energy expenditure, and the term p^R is not needed for the source node. A leaf node, since it does not transmit but only receives, has a total power expenditure of p^R. We assume that each node starts with a finite quantity of battery energy. For example, Node *i* has energy $E_i(0)$ at time 0. The *residual energy* at Node *i* at time *t* is

$$E_i(t) = E_i(0) - \int_0^t P_i(\tau)d_\tau \quad (7)$$

Where $P_i(\tau)$ is the total power expended at Node *i* at time τ. We say that a node is "alive" as long as its residual energy is positive and that it dies when its residual energy decreases to zero. Based on our assumptions, a "dead" node cannot participate, even as a leaf node.

5.2.3 Proposed Algorithm

5.2.3.1 Localized Energy Aware Broadcast Algorithm

The goal of the localized algorithm [12] is to allow a localized and distributed computation of broadcast tree. We assume every node knows the position information of its neighbors within two hops.

The principle is as follows: the source node *S* (the one that initiates the broadcast) computes the broadcast tree within its two hops neighborhood, and sends the broadcast packet to each of its one hop neighbor, while the determined relay instructions for relay

nodes are included in broadcast packet. For each of its one hop neighbors, for example, node *U* who receives the packet for the first time, two cases can happen:

- The packet contains relay instructions for *U*. *U* will use these relay instructions to construct its own two hop broadcast tree. Then, instead of starting from an empty tree as *S* did, it extends the broadcasting tree based on what source *S* has calculated for it. By this way, the joint neighborhood nodes of *S* and *U* will use the same spanning tree.

- There are no relay instructions for *U*. In this case, starting from an empty tree, node *U* can just use its two hops neighbor location information to construct its own broadcast tree.

After the procedure mentioned above, node *U* will rebroadcast the packet again to its own one hop neighbor and includes relay information for its own relay node, just like what source node has done.

In this principle, there may be some nodes which will receive this packet more than one time, then at this time, node can simple drop the packet and doesn't rebroadcast again. In order to reduce overlap, we use the neighbor nodes elimination scheme. Each node will include its two hops neighbor nodes in packet, because these nodes certainly will receive the packet soon. Once a node receives the packet, except recording the relay information, it should also record the nodes which will be covered soon. If the covered node is not used in relay information and also is a neighbor node of this node, then this node will delete it from its neighbor list and after deletion calculate its own broadcast tree. Fig.2.2 is the pseudo-code of the proposed algorithm.

As for how to set up broadcast tree, we have considered two basic approaches with directional antennas:

- Construct the tree by using an algorithm designed for omni-directional antennas; then, reduce each antenna beam to our fixed beam width.

- Incorporate directional antenna properties into the tree-construction process.

```
0.  Randomly select source node S
1. For source node S:
2.  {   /******source node's localized calculation******/
3.     Computes the broadcast tree within its two hops neighborhood;
4.     Set up broadcast packet P;
5.     Include relay instructions in packet P;
6.     Include all neighbors' ID in packet P;
7.     Send packet P to each of its one hop neighbor using directional antenna;
8.  }
9.  For any node U (except S):
10. if (node U receives packet P){
11.     if ( the first time){
12.        Search and record relay instructions for U;
13.        /******Neighbor Nodes Elimination Scheme******/
14.        Check included nodes' ID;
15.        While ( (ID != U's address)&&( ID∉ relay instruction info) ){
16.           if (ID⊂U's two hops neighbors list)
17.              delete this node record from U's two hops neighbors list;
18.        }
19.        /******U's localized calculation******/
20.        Refer recorded relay instructions;
21.        Use U's modified two hops neighbors list;
22.        Computes U's broadcast tree within its two hops neighborhood;
23.        Act as source node;
24.     } else
25.        Simply drop packet P;
26. }
```

Figure 2.2: Localized Broadcast Algorithm

The first approach can be based on any tree-construction algorithm. The "beam-reduction" phase is performed after the tree is constructed The second approach, which takes directional antenna into consideration at each step of the tree construction process, can be used only with algorithms that construct trees by adding one node at a time. In next section, we describe the later approaches (LDBIP) in detail.

5.2.3.2 Localized Directional Broadcast Incremental Power Protocol (LDBIP)

The incremental power philosophy, originally developed for use with omni-directional antennas, can be applied to tree construction in networks with directional antennas as well. At each step of the tree-construction process, a single node is added, whereas variables involved in computing cost (and incremental cost) are not only transmitter power but beam width θ as well. In our simple system model, we use fixed beam width θ_f, that means for adding a new node, we can only have two choices: set up a new

directional antenna to reach a new node; raise the length range of beam to check whether there is new node covered or not.

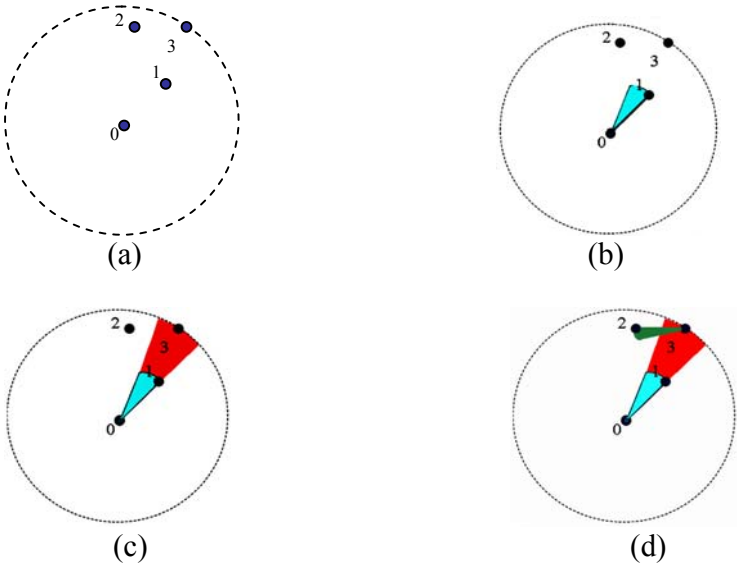


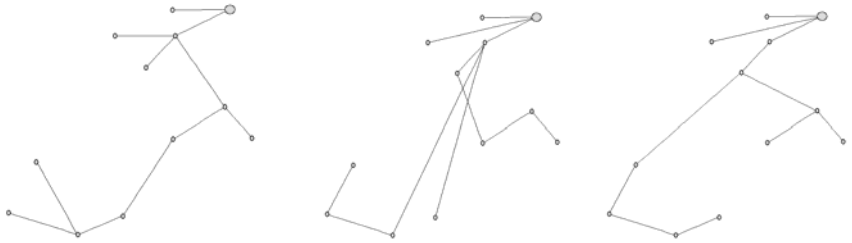
Figure 2.3: Nodes addition in LDBIP

Fig. 2.3(a) shows a simple example in which the source node 0 wants to add nodes 1, 2, and 3 to the tree. Node 1 is the closest to 0, so it is added first; in Fig. 2.3(b), an antenna with beam width of θ_f is centered between 0 and Node 1. Then we must decide which node to add next (Node 2 or Node 3), and which node (that is already in the tree) should be its parent. In this example, the beam from 0 to Node 1 can be extended to include both Node 1 and Node 3, without setting up a new beam. Compared to other choices that setting up a new beam from Node 0 to Node 2, or from Node 1 to Node 2, this method has minimum incremental power. Therefore, Node 3 is added next by increasing the communication range of 0 and Node 1. In Fig. 2.3(c), finally, Node 1 must be added to the tree. Three possibilities are respectively to set up a new beam from 0, 1, 3. Node 3 has minimum distance. Then in Fig. 2.3 (d) we set up a new beam from Node 3 to Node 2.

Once calculate the two hops neighborhood broadcast tree, source node will send packets to its one hop neighbors. In above example, Node 1 and Node 3 are Node 0's one hop neighbors. Once they receive the packet, they will check the relay information and covered node information. For example, as for Node 1, it doesn't receive relay information, and it has only three neighbors, which are Node 0, 2 and 3. However, packet originated from Node 0 includes all of them as covered nodes. Then Node 1 has no neighbor nodes to set up its own broadcast tree, so it will give up rebroadcast. As for Node 3, we assume it also has other two hops neighbors except Node 1 and 2. Because Node 3 received the relay information to 2, even Node 2 is included in covered nodes list, it still can participate in the new broadcast tree construction. Furthermore, when Node 3 constructs its own broadcast tree, it has to add the link of Node 3 to Node 2 as its first downlink beam as Node 0 has calculated. By this way, even every node only knows its two hops neighbors and only calculates its two hops broadcast tree, packets still can be

broadcasted in the whole network. In addition, when any changes of sensor nodes in their activity status occur, with localized broadcast protocol, we can get the minimum computation and energy consumption compared to centralized solutions. They are the advantages of our algorithms.

5.2.3.2 Examples Constructed by the Various Algorithms



(a) (b) (c)

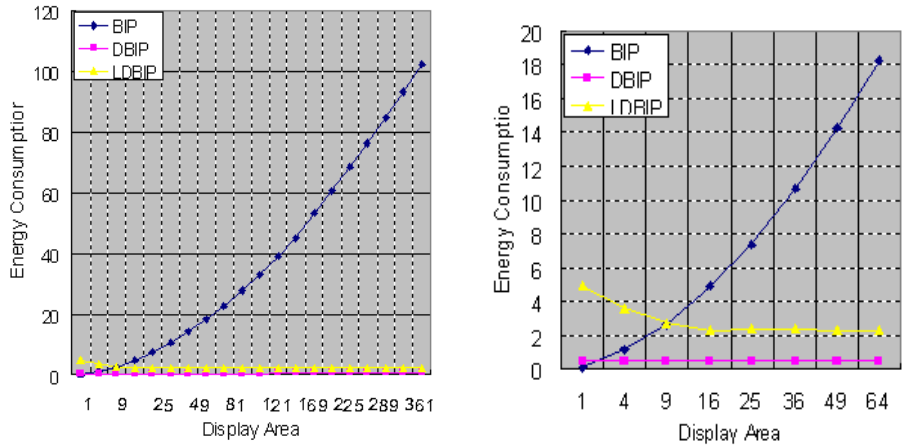
Figure 2.4: Broadcast Tree. (a) BIP (b) DBIP(θ_f =30) (c) LDBIP(θ_f =30)

Fig. 2.4 shows the broadcast tree produced by BIP, DBIP and LDBIP for a 12-node network, where the source node is shown larger than the other nodes. There broadcast trees are generated in our simulation work, which use the system and energy model mentioned in section 2.

Because DBIP and LDBIP use directional antenna, therefore in our simulation system, according to different θ_f, we can get different broadcast tree; of course, the according energy consumption will also be different. Furthermore, because our algorithm LDBIP is distributed, which means every node only calculates its two hops neighborhood broadcast tree, the Fig. 4 (c) in fact is the combination of all nodes' broadcast tree. Based on our algorithm, the joint parts of nodes' broadcast tree will not have too much difference because nodes refer relay information from other nodes and apply the neighbor nodes elimination scheme.

5.2.4 Performance Evaluation

In this section, we present our performance evaluation for our localized algorithm *LDBIP*, and also compare it with two centralized algorithm *BIP* and *DBIP*, since these two protocols are very effective centralized protocols in energy consumption.



(a) (b)

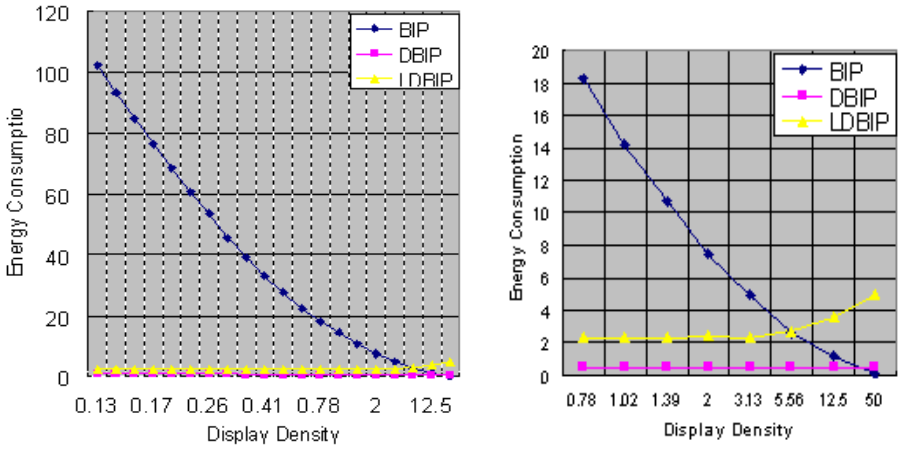
Figure 2.5: Energy Consumption Comparison

We use ns2 as our simulation tool and apply the system, propagation, and energy model mentioned in Section 2. The parameters used in our simulations are the following. The network is static and always composed of 50 nodes randomly placed in a square area whose size is changed to obtain different density. For each measure, 5 broadcasts are launched and for each broadcast, a new network is generated.

To compare the performance with those of other protocols, we observe the total power consumption over the network when broadcasts have occurred.

Fig.2.5 shows the relationship of energy consumption and display area for BIP, DBIP and LBIP protocols. Fig.2.5 (a) shows the comparison in a large display area range and Fig.2.5 (b) focuses on the small display area.

Fig.2.6 shows the relationship of energy consumption and display density for *BIP, DBIP* and *LBIP* protocols. Fig.2.6 (a) shows the comparison in a large display density range and Fig.2.6 (b) focuses on the high display density.



(a) (b)

Figure 2.6: Energy Consumption Comparison

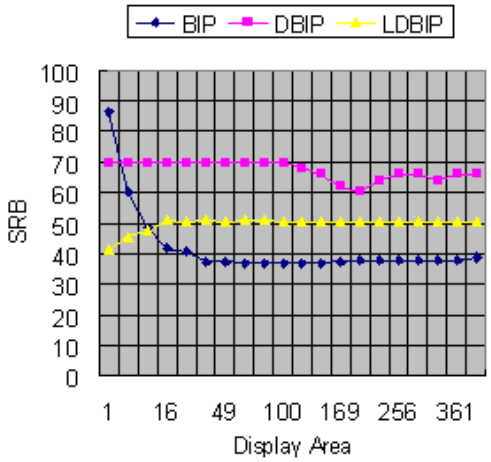


Figure 2.7: Saved Rebroadcast Ratio

From Fig.2.5 and 2.6, we can find that because of using directional antenna, DBIP and LDBIP have stable energy consumption despite of display area and density and are very good at energy saving. Compared to centralized algorithm DBIP, in wireless network, our localized algorithm LDBIP has a little more energy consumption. However, this is negligible because our algorithm needs the topology of only two hop neighbors whereas DBIP requires the total network topology.

We also observe the value of *SRB* (*Saved Rebroadcast*) which is the percentage of nodes in the network that received the message but did not relay it. A Blind Flooding has a *SRB* of 0%, since each node has to retransmit the message once.

Fig. 2.7 shows the relationship of SRB and display area for *BIP, DBIP* and *LBIP* protocols. From this diagram, we can see compared omni-directional antenna, using directional antenna we can get higher SRB value despite of display area. Our localized protocol LDBIP has higher SRB compared to BIP, but has less SRB compared to DBIP. However, our localized algorithm has more stable SRB compared to BIP, even DBIP. From above diagram, we can see that basically the SRB of LDBIP is 50%.

In summary, in wireless networks, our localized energy-aware protocol has good performance in energy consumption and saving rebroadcast, which is similar to well-known centralized protocol DBIP. Since our protocol is based on localized information and distributed computation, if consider topology changes, nodes only need broadcasting them to their neighbors rather than all the other nodes in the whole network, our localized protocol can get better performance.

5.2.5 Conclusions

In this chapter, we proposed the new localized energy-aware broadcast protocol for wireless networks with directional antenna which has limited energy and computation

resources such as sensor network. Our algorithm is based on the localized information and distributed computation method, which means, rather than source node collects all location information of network to calculate broadcast tree, every node collects some part of the whole network's nodes location information and participates calculating broadcast tree. At the cost of a few more information stored in the broadcast packets, our localized algorithm offers near as good energy saving result as well-known centralized algorithm DBIP. Its drawback is that since every node calculates the broadcast tree only based on part of all location information, so compared to centralized algorithm, the power consumption for total spanning tree may be not the minimum. However, with localized information, nodes can broadcast in a distributed way, and don't need to collect all the location information in network.

Especially, if consider topology changes caused by nodes' status changes, our distributed algorithm can get the minimum recalculating computation and in this aspect can get lesser energy consumption. So in future work, we plan to take topology changes into consideration for our simulation. In addition, we will take realistic facts into consideration for energy consumption and network lifetime.

5.2.6 References

[1] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides, On the construction of energy-efficient broadcast and multicast trees in wireless networks, Proc. IEEE INFOCOM 2000, pp. 585-594, March 2000.
[2] J.E. Wieselthier, G.D. Nguyen, and A. Ephremides, Energy-Limited Wireless Networking with Directional Antennas: The Case of Session-Based Multicasting, Proc. IEEE INFOCOM 2002, pp. 190-199, June 2002.
[3] P. Bose, P. Morin, I. Stojmenovic, J. Urrutia, Routing with guarantee delivery in ad hoc networks, CM/Kluwer Wireless Networks 7 (6) (2001) 609-616.
[4] T. Chu, I. Nikolaidis, Energy efficient broadcast in mobile ad hoc networks, in: Proc. Ad-Hoc Networks and Wireless (ADHOC-NOW), Toronto, Canada
[5] W. Peng, X. Lu, On the reduction of broadcast redundancy in mobile ad hoc networks, in: Proc. Annual Workshop on Mobile and Ad Hoc Networking and Computing (MobiHoc'2000), Boston, Massachusetts, USA, 2000, pp. 129-130.
[6] A. Qayyum, L. Viennot, A.Laouiti, Multipoint relaying for flooding broadcast messages in mobile wireless networks, in: Proc. 35th Annual Hawaii International Conference on System Sciences (HICSS-35), Hawaii, USA, 2002.
[7] J. Wu, H. Li, A dominating-set-based routing scheme in ad hoc wireless networks, in: Proc. 3rd Int'l Workshop Discrete Algorithms and Methods for Mobile Computing and Comm (DIALM'99), Seattle, USA, 1999, pp. 7-14.
[8] H.T. Friis, A note on a simple transmission formula, Proc. of the IRE, Vol. 41, May 1946, pp. 254-256.
[9] H.T. Friis, Introduction to radio and radio antennas, IEEE Spectrum, pp. 55-61, April 1971.
[10] Joseph J. Carr, "Directional or Omni-directional Antenna" Joe Carr's Receiving Antenna Handbook, Hightext, 1993.

[11] J.E. Wieselthier, G.D. Nguyen, and A. Ephremides, Algorithms for Energy-Efficient Multicasting in Static Ad Hoc Wireless Networks, Mobile Networks and Applications (MONET), vol. 6,no. 3, pp. 251-263, June 2001.
[12] F.Ingelrest, D.Simplot-Ryl, Localized Broadcast Incremental Power Protocol for Wireless Ad Hoc Networks. 10th IEEE Symposium on Computers and Communications (ISCC 2005), Cartagena, Spain, June 2005.

5.3 Swarm Intelligence Inspired Autonomic Routing

5.3.1 Introduction

There is a need to manage heterogeneous computing devices in accordance with high level policies set out by their human administrators [1]. This is particularly the case for devices in distributed ubiquitous environments. This autonomous style of management is referred to as autonomic computing [2] which includes at least four basic characteristics, namely self-configuration, self-optimization, self-healing, and self-protection. The purpose is that autonomic computing system will make the job of human administrator more relaxed, since the complexity of managing heterogeneous computing units is getting too large to be handled in current scenarios.

A novel alternative approach to reduce system complexity is to make use of the Swarm Intelligence (SI) mechanism. A SI inspired system is a self-optimizing system with four main ingredients, which are positive feedback, negative feedback, amplification of fluctuations and multiple interactions among multi-agents [9]. Strangely, it also covers the four aspects of autonomic computing to some degree. In SI inspired systems, many insect like agents are interacting locally with one another and with their environment, just like the relationship between autonomic elements and their managing environment, and finally some global optimized performance can be achieved, such as to find the shortest path from the nest to the forage place. Usually, the agents are simple and small in size, just like the insects which have only few hundreds or thousands of neurons. So, the software packets can also be small in size. Also, the SI inspired systems are robust and the tasks can be fulfilled through the interaction among the multiple agents with faster convergence. In the next section, a detailed comparison between the SI inspired mechanism and autonomic computing will be presented.

Due to the inherent accordance between autonomic computing and swarm intelligence, it is reasonable to combine them together and to apply them to some distributed and ubiquitous environment, such as the Ubiquitous Sensor Network (USN). To the best of our knowledge, little work has been done to combine the mechanism of SI with autonomic computing and then to apply them to some ubiquitous applications.
Our contribution in this chapter lies in the following three aspects. First, we explore the inherent accordance between autonomic computing and swarm intelligence mechanism. Then, we propose a SI inspired autonomic routing scenario in a ubiquitous context aware

application environment. Finally, we present our preliminary simulation results and analysis of some performance metrics, such as agent number, packet delivery ratio and power consumption.

5.3.2 Comparison between Autonomic Computing and Swarm Intelligence

Autonomic computing [1-3] system is a self-managed computing system with minimum human consciousness or involvement, and with an ultimate goal to free administrators from the burden of system operation and maintenance. It has at least four basic ingredients, which are self-configuration, self-optimization, self-healing, and self-protection. Different aspects of autonomic computing are explained in [4-6] with specific application emphasis. Opportunities and possible research directions of autonomic computing in the system engineering field are well explained in [7], and [8] gives a concise and comprehensive introduction about autonomic computing.

The principles of Swarm Intelligence (SI) [9] were originally inspired by the observation of various natural and social phenomena. Even though these social insects have only few hundreds or thousands of neurons and they simply collaborate with their neighbours, some high level intelligence is observed. Thus, the idea that subcomponents are very simple while the overall system manages itself adaptively is very tempting and promising from an engineering perspective. Ant colonies apply this principle, which has inspired a new research direction called Ant Colony Optimization (ACO) [10].

The SI inspired system is a self-organizing system, which includes positive feedback, negative feedback, amplification of fluctuations and multiple interactions among multi-agents [9]. The relationship between autonomic computing and swarm intelligence is tightly coupled according to Table 1. The middle column in Table 1 is a concise interpretation which bridges both autonomic computing and swarm intelligence. For example, through the tuning of system parameters, the self-configuration aspect can be achieved autonomically rather than manually. In the mean time, the pheromone can get evaporated, reinforced, thresholded according to [9], which represents the actual parameters under dynamic system situation. From this, we can see that both of them are tightly coupled. It is obvious that the system performance can get optimized through the local interaction among the multiple ant agents. Also, the system robustness can be guaranteed through this kind of parallel multi-agent interaction. Finally, the security and load-balancing can be fulfilled with careful parameter and rule design. In the next section, the readers can get a deeper understanding of their inherent accordance.

Table 1. Relationship between autonomic computing and swarm intelligence

Autonomic Computing	Relationship	Swarm Intelligence
Self-Configuration	System parameters tuning etc.	Pheromone based operator
Self-Optimization	Optimized system performance etc.	Global optimal performance through local interaction
Self-Healing	Robustness, convergence rate etc.	Multi-agent interaction, (parallelism)
Self-Protecting	Security, load-balancing etc.	Pheromone threshold, sequence number etc.

5.3.3 A-CAMUS Architecture

Our work in this chapter is influenced by the earlier work relating to the Autonomic Context-Aware Middleware for Ubiquitous Systems (A-CAMUS) architecture for ubiquitous sensor networks [11-13]. This project been undertaken for more than 3 years and a "Smart Office" test-bed has been built. We include a brief discussion about A-CAMUS here as background and related work.

In Fig.1 lists our A-CAMUS architecture.

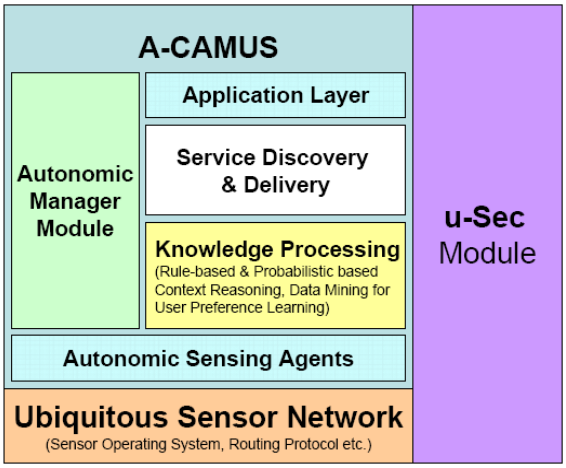


Fig. 1. An A-CAMUS Architecture

The ubiquitous sensor network lies in the bottom layer. Some sensing devices, such as Berkeley-mote, RFID devices etc. are deployed in the real environment, which form a ubiquitous sensor network. These devices can collect the original information and then send this raw data to the upper knowledge processing layer for further processing with the help of autonomic sensing agents, which is defined as software. During this process, routing is an utmost important task since it should not only send the related data from the source to the destination correctly, but that it should meet the constraints and provide certain QoS requirement. In this chapter, we try to solve this routing problem by using swarm intelligence mechanism.

5.3.4 Swarm Intelligence inspired Autonomic Routing

Based on the inherent accordance between autonomic computing and swarm intelligence, we now provide a SI inspired autonomic routing scenario (which is a sub-component of A-CAMUS architecture).

5.3.4.1 Routing Scenario

Fig. 2 shows three modules of our agent based autonomic routing scenario which are Autonomic Agent (AA) manager, Service Manager and Context Manager. We will add more functionality into the modules and verify it based on our "Smart Office" test bed later.

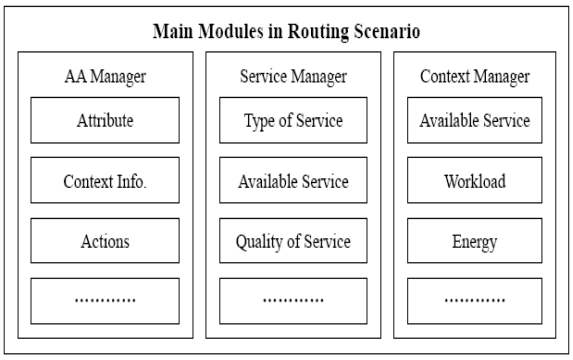


Fig. 2. Autonomic routing scenario

TABLE 2 DATA STRUCTURE OF AA MANAGER

Attributes	Unique ID, Timestamp, Remaining energy, Service type etc.
Context Information	IP, Available service, Hop number, Pheromone distribution, Humidity, Temperature, Lighteness, latency, etc
Actions	Generation, Migration, Service discovery, Pheromone evaporation, Pheromone aging, Pheromone reinforcement, etc.

Autonomic Agent (AA) manager consists of three main parts, which are attributes, context information and actions, as is shown in Table 2. Autonomic Agents (AAs) will collaborate with each other and configure themselves so as to achieve an optimal performance based on SI mechanism. While in the mean time, they will prevent the system from being over-loaded and malfunctioned to fulfill the self-protecting function. A Service Manager is in charge of different services, like routing, printing services etc. Then, through the communication among AAs, different types of services can be collected and exchanged in a localized and real-time manner. A Context Manager is like a repository to store the context and provide the best service so as to fulfill the self-optimization function.

An SI inspired routing scenario based on these three modules is proposed in Fig. 3. From which, we can conclude its basic steps as follows:

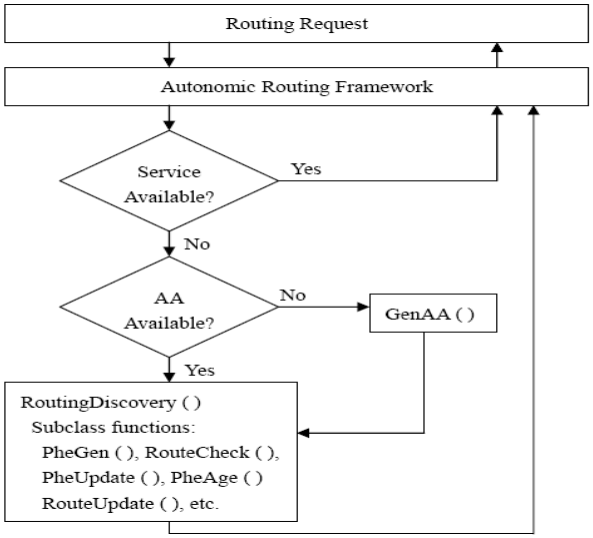


Fig. 3. Autonomic routing work flow

Step 1: Originally, each AA is piggybacked into the hello packet and periodically broadcasted to collect some information, such as available services, remaining energy, distance, next hop etc. The SI mechanism is used with pheromone evaporation, reinforcement and aging.

Step 2: When there is some specific routing request originated from one node, it will follow the autonomic routing work flow in Fig. 3, which consists of many closed control loop, as is the same with the autonomic computing.

Step 3: Then, it will first check the Service Manager in Fig. 2 to see whether the service is available, namely whether the destination node is in its routing table. If it is available, it will check the context manager to get the best candidate and return it to the original node finally.

Step 4: If the service is not available, it will contact the AA Manager rather than search in the Context Manager. If AA is attacked or malfunctioned, it will autonomically generate another one with the uniform data structure shown in Table 2. In this way, a self-healing function is fulfilled here. If AA is available, the service discovery process will be initiated so as to find the required service. Through the localized interaction between AAs, the related services can be found, like to find the shortest path between source node and destination node. During this process, some actions are taken, such as pheromone generation, updating, and routing updating etc. to fulfill the self-configuration function.

Step 5: Once specific route is found, the context information is sent back to the Service Manager and Context Manager. And finally, the best service is decided and provided to the users. Here, another closed control loop is formed.

5.3.4.2 Routing Phase

The routing phrase can be divided into three phases, which are route setup, route maintenance and route failure handling phase. Here, we will illustrate this procedure with an example depicted in Fig. 4.

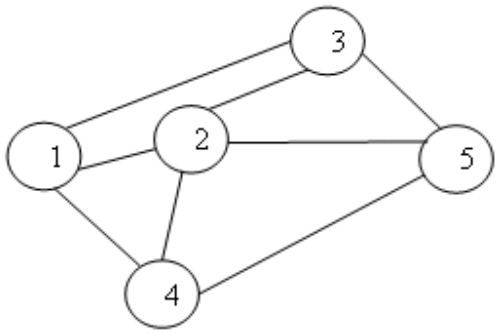


Fig. 4. 5 nodes network topology

Table 3 Pheromone table of node 1

	2	3	4	5
2	$p_{2,2}$	$p_{2,3}$	$p_{2,4}$	$p_{2,5}$
3	$p_{3,2}$	$p_{3,3}$	$p_{3,4}$	$p_{3,5}$
4	$p_{4,2}$	$p_{4,3}$	$p_{4,4}$	$p_{4,5}$
5	$p_{5,2}$	$p_{5,3}$	$p_{5,4}$	$p_{5,5}$

During the route setup phase, each node will select its next hop node to forward the traffic packet according to its routing table and pheromone table. Taking node 1 as an example, its pheromone table is shown as in Table 3. The row in pheromone table represents its neighboring neighbors and column represents the destination node. The entry means the probability to get to the destination node through the neighboring nodes. Once node 1 has packet to send to node 5 and it has no pheromone information about node 5, it will initiate route setup phase. If the intermediate node has no information about destination node, it will continually broadcast until the broadcast packet reaches the destination node. Finally, the route is setup on reverse direction. It is worth noting that, the pheromone value is only updated during the backward route setup phase so as to represent the latest update of dynamic network. The selection of the probability entry $p_{i,j}$ can be found in [9].

During the route maintenance phase, the source-destination pair is maintained either through periodical hello packets or data packets to save routing overhead. In the meantime, the pheromone value is either reinforced or evaporated to represent the real network situation. Since the selection of next hop is probability based, the work load on

one route can be shared by the other route to make load-balancing. In this way, the functionalities of self-configuration, self-optimization and self-protecting are achieved.

Finally, once there is a link failure due to overload or out of power, the local repair can be adopted so as to search for alternative route. If impossible, a new route setup phase will be initiated.

5.3.4.3 Energy Consumption Model

Here, we use one of the energy consumption model from [16] as below. The energy needed to transmit, receive and forward n_b bits of packets are calculated in (1), (2) and (3) respectively.

$$E_T = n_b(E_{elec} + E_{amp} \times d^{\alpha}) \quad (1)$$

$$E_R = n_b \times E_{elec} \quad (2)$$

$$E_F = E_R + E_T = n_b(2E_{elec} + E_{amp} \times d^{\alpha}) \quad (3)$$

Here, $E_{elec} = 50nJ/bit$, $\alpha =$ 2 or 4. If the distance d is smaller than a distance threshold d_T, a free space fading (d^2 power loss) model is adopted and $E_{amp} = \varepsilon_{fs} = 10pJ/bit/m^2$. Or else, a multipath fading (d^4 power loss) model is adopted and $E_{amp} = \varepsilon_{mp} = 0.0013pJ/bit/m^4$. Here, we let:

$$\varepsilon_{fs} \cdot d^2 = \varepsilon_{mp} \cdot d^4. \quad (4)$$

So, we can get the distance threshold (d_T) as 87.7 meters.

5.3.4.4 Study of Energy Consumption

Now, we will study the factors which influence the energy consumption from source node to the destination node. Taking Fig. 1 as an example, there are N=30 nodes randomly deployed in a [1000, 1000] m^2 area. Let $d_{AB} = A$, $d_{BC} = B$, $d_{AC} = C$ and $A \le B < C$ (also it could be $B \le A < C$ since A and B are replaceable here). We simply put n_b as one bit and we can easily calculate the bit number according to the traffic model in real situation.

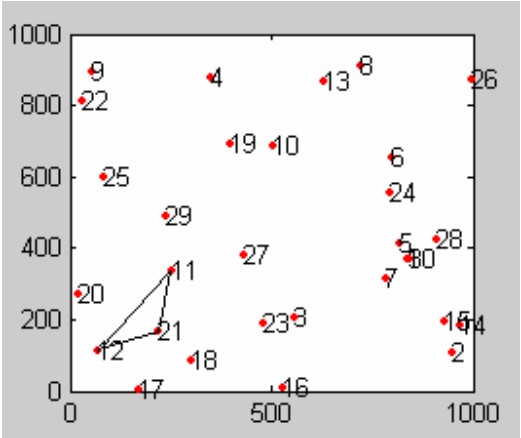


Fig. 1. 30 nodes deployment

First, we will study the one hop instance, where the relationship between $E_A + E_B$ and E_C is studied. Then, we can extend it to multi-hop circumstances. We average our simulation results for 10 times and here, we simply provide one instance so as to make it intuitively easy to understand, as is shown in Table 3.

Table 3. Comparison of one hop energy consumption (J)

Case 1: $C<d_T$	$E(2,15)= 1.79*10^{-7}$
A, B< d_T	$E(2,14)+E(14,15) =2.78*10^{-7}$
Case 2: $C>d_T$	$E(11,12)=9.42*10^{-6}$
A, B> d_T	$E(11,21)+E(21,12) =2.17*10^{-6}$
Case 3: $C>d_T$	$E(7,5)= 2.64*10^{-7}$
A, B< d_T	$E(7,1)+E(1,5) =2.82*10^{-7}$
Case 4: $C>d_T$	$E(3,27)=2.88*10^{-6}$
A< d_T<B	$E(3,23)+E(23,27) =2.38*10^{-6}$

Table 3 lists 4 cases about the energy consumption between $E_A + E_B$ and E_C according to their special position of node A, B and C. In Case 1, the distance between node A and C is smaller than the distance threshold d_T. It is an ideal situation since we can simply communicate from node A to node C without the help of any intermediate nodes. The energy consumption is in the order of 10^{-7} and it is also less than the summation from multi-hop nodes. In Case 2, the distance between Node 11 and Node 12 is C=291 meters, and A=159 meters, B=172 meters. From equations (1)-(3), we obtain $E_C = 9.4*10^{-6}$, which is about four times larger than $E_A + E_B$. It is worth noting that the energy consumption is in the order of 10^{-6} to transmit only one bit data.

We can see that if A and B are both smaller than the distance threshold d_T, the difference between $E_A + E_B$ and E_C is small. Usually, we can neglect it and take either of the instances. If one of the intermediate distance is larger than d_T, the difference between $E_A + E_B$ and E_C is larger than the first case. The difference degree is about ten times larger. And if all the distances are larger than d_T, the difference is the largest among all the cases. In Case 3, C is larger than d_T while A and B are both smaller than

d_T. We can see that E_C is smaller than $E_A + E_B$ and the difference between them is also small. Finally, in Case 4, we observe that there are situations when A<d_T<<B, it could be better to transmit the data through multi-hop nodes rather than to transmit the data through one-hop.

So, here, we can draw one conclusion that the difference between $E_A + E_B$ and E_C will not be very large if A and B are not much larger than the distance threshold. However, if one of the distance or both A and B are larger than the threshold, it is a wise choice to use multi-hop transmission. Now, we can extend this conclusion to the multi-hop instance and try to validate it.

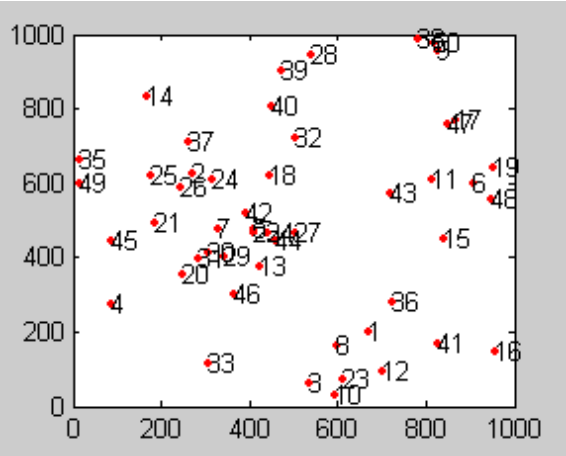


Fig. 2. 50 nodes deployment

Next, we will increase the network connectivity and introduce 50 nodes deployment in a similar way as Fig. 1, which is shown in Fig. 2. We will compare the energy consumption in the context of multi-hop routing during one communication traffic.

From Table 4, we can draw a similar conclusion as in Table 3, which is that the energy consumption will not be too large if the communication distance is not much larger than the distance threshold. In other words, if we choose those nodes with communication distance smaller or a little larger than the distance threshold, the energy consumption is reasonably acceptable. So, we are willing to save much energy at the cost of a few more hop numbers. Once again, it validates our selection criterion about the intermediate nodes during the routing phase.

Table 4. Comparison of multi-hop energy consumption (J)

Case 1: C>d_T		E(20,29)+E(29,22)= $2.47*10^{-7}$
A, B..<d_T		E(20,31)+E(31,30)+E(30,29)+ E(29,22) =$5.35*10^{-7}$
Case 2: C>d_T		E(26,42)+E(42,46)= $4.14*10^{-6}$
A< d_T<B	..<	E(26,24)+E(24,7)+E(7,30)+E (30,29)+E(29,46)=$1.14*10^{-6}$
Case 3: C>d_T		E(43, 36)+E(36, 16)= $1.60*10^{-5}$
A, B..>d_T		E(43,11)+E(11,15)+E(15, 36)+ E(36,41)+E(41, 16)=$4.95*10^{-6}$

5.3.5 Experimental Setup and Results

5.3.5.1 Experimental setup

Based on our previous experience [14], we set our simulation environment as follows. There are N nodes randomly placed within a range of 100 by 100 m^2, with a uniform transmission range R of 30m. A Random Waypoint mobility model is adopted here and their velocity vary from 0 to 20 m/s. Taking N=20 as an example, we will see the deployment as in Fig.5.

Unlike the broadcasting mechanism, we set our neighbor selection criteria as follows:

$$p_{ij}(t) = e^{-\tau_i(t)} * e^{-d_{ij}/R}, \quad (d_{ij} < R) \tag{2}$$

$$\tau_i(t+1) = \tau_i(t) + \Delta(t, t+1), \ \tau \in [0,1] \tag{3}$$

$$\Delta(t,t+1) = \begin{cases} K * \dfrac{1}{2 \cdot N} & reinforced \\ (\rho - 1) * \tau_i(t) & aging \end{cases} \tag{4}$$

here, $\tau_i(t)$ is the pheromone distribution of node i at time t, d_{ij} is the distance of <i, j> and R is the maximum transmission range. From (2), we can infer that the smaller d_{ij} and $\tau_i(t)$ is, the higher $p_{ij}(t)$ will be and we will choose higher $p_{ij}(t)$ as the candidate node. In other words, this algorithm will prefer those nodes which are either nearer to the transmission node or being less visited with less pheromone so as to save energy. After each hop, the pheromone will automatically be updated according to (3) and (4), where the pheromone will either be reinforced according to the number of visiting agents K or aged by a factor of ρ (normally between [0.9, 1]) and N is the neighbor number.

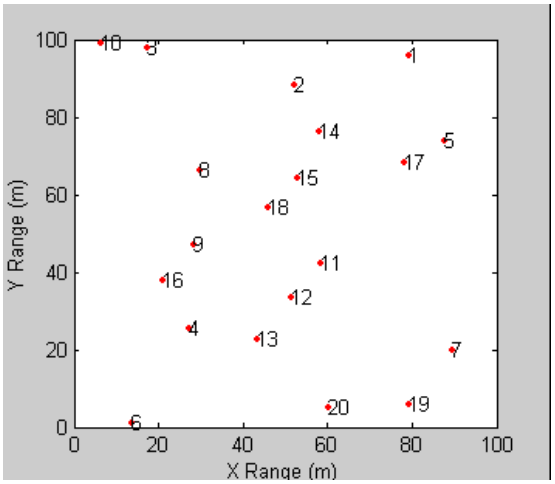


Fig. 5. N (20 in this case) node random deployment

5.3.5.2 Study of Agent Number

We will first study how many autonomic agents is need for each node so as to fulfill the routing task. If the agent number is too small, it will not find the related route or the response time will be too large to accept. If the agent number is too large, the routing overhead will be a burden and it is similar to the broadcasting mechanism, which we try to avoid in this chapter.

Table.4 shows the average number of neighbors for each node. The total node number varies from 20 to 100 and the average node neighbor increases almost linearly with it. So, we tentatively set our agent number as K=**min** (N, 6) here and we will try to validate it later.

TABLE 4 AVERAGE NEIGHBOR FOR DIFFERENT N

N	10	20	30	40	50
Neighbor	2	4	6	8	10
N	60	70	80	90	100
Neighbor	12	14	17	20	22

5.3.5.3 Study of Packet Delivery Ratio

To verify the agent number, we randomly generate 300 routing connections between node *i* and *j*. In Fig. 6, we make a comparison between one agent approach and our approach in the aspect of packet delivery ratio. From this figure, we can see that there is no reliability guarantee if the agent number is too small. For the connection of (1, 14) and the node deployment in Fig. 5, if there is only one agent, the routing sequence will be {1, 5, 15, 6, 20, 18, 17, 13, abort}. If there are K agents like our approach, it will first take nodes {5, 16} as its first hop node. Then, node 5 will detect node 14 as the destination node and return this information to the source node 1. It only takes 2 hops. And another

routing sequence can also be found through {1, 16, 19, 18, 9, 8, 14}. So, the successfulness of finding a reliable route can be ensured in our approach.

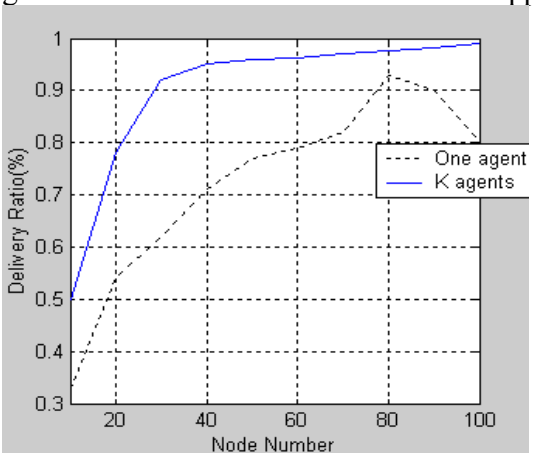


Fig. 6. Packet delivery ratio for different approaches

5.3.5.4 Study of Power Consumption

Finally, we will study the power consumption and make a comparison between the broadcasting mechanism and our approach. According to our previous work in [15], we assume the power consumption is proportional to the square of distance and the data length is the same.

Fig.7 shows the power consumption for different approaches, from which we can see that our approach is always better than the broadcasting mechanism. And the larger N is, more energy will be saved. So, the self-optimization aspect can be once again achieved.

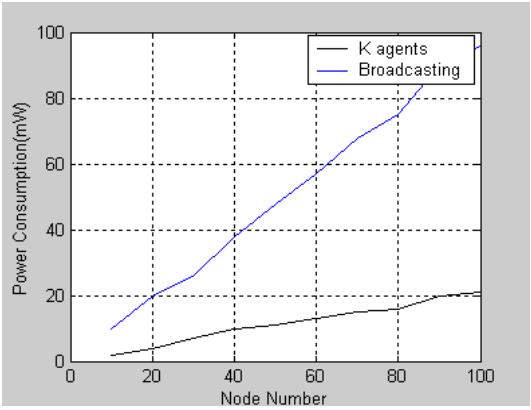


Fig. 7. Power consumption for different approaches

5.3.6 Conclusion and Future Work

We compare and draw some inherent accordance between autonomic computing and swarm intelligence mechanism. Then, we introduce a Swarm Intelligence inspired autonomic routing scenario under a context-aware ubiquitous environment. The autonomic routing scenario and routing phase are explained in detail, and some preliminary experimental results are provided with satisfactory performances. The energy

consumption about SI inspired routing algorithms is studied afterward based on one famous energy model. From the experiments of both one hop and multi-hop instances, we provide an intermediate node selection criterion which is quite energy-efficient.

For the present, an agent based "Smart Office" test bed has been built in our professor's office. With the aid of various sensors, actuators and protocols, we intend to perform extensive experiments based on our SI inspired autonomic routing scenario, and introduce more modules and functionalities in the near future.

5.3.7 References

[1] R. Murch, "Autonomic Computing," Upper Saddle River, NJ: Prentice Hall, Mar. 2004, ISBN: 013144025X.

[2]IBM, "Autonomic Computing Initiative," IBM Press, 2003, http://www.autonomic-computing.org.

[3] J. Kephart and D. Chess, "The Vision of Autonomic Computing," IEEE Computer, vol. 36, no. 1, January 2003.

[4] Belecheanu, R. A., Jawaheer, G., Hoskins, A., McCann, J. and Payne, T, "Semantic Web Meets Autonomic Ubicomp," In Proceedings of The 3rd International Semantic Web Conference, Hiroshima, Japan, 2004.

[5] F. Schintke, T. Schütt, A. Reinefeld, "A Framework for Self-Optimizing Grids Using P2P Components", 14th Intl. Workshop on Database and Expert Systems Applications (DEXA'03), September 2003, pp. 689 – 693.

[6]Apostolos Malatras, George Pavlou, et al, "Self-Configuring and Optimizing Mobile Ad Hoc Networks," Proceedings of the Second International Conference on Autonomic Computing (ICAC'05), Seattle, Washington, June 2005, pp. 372-373.

[7] H. Schmeck, "Autonomic computing - vision and challenge for system design," in Proceedings of the International Conference on Parallel Computing in Electrical Engineering (PARELEC'04), Dresden, Germany, September 2004.

[8] R. Sterritt, M. Parashar, H. Tianfield and R. Unland, "A Concise Introduction to Autonomic Computing," Journal of Advanced Engineering Informatics, Special Issue on Autonomic Computing and Automation, Elsevier Publishers, 2005, pp. 181-187.

[9] E. Bonabeau, M. Dorigo, G. Theraulaz, "Swarm Intelligence: From Natural to Artificial Systems", Oxford University Press, New York, 1999.

[10] Maniezzo V, Gambardella LM, De Luigi F, "Ant Colony Optimization, New Optimization Techniques in Engineering," by Onwubolu, GC, and BV Babu, Springer-Verlag Berlin Heidelberg, 2004, pp. 101-117.

[11] Hung Q. Ngo, Hung Le-Xuan, SungYoung Lee. A Middleware Framework for Context Acquisition in Ubiquitous Computing Systems. Second International Conference on Computer Applications (ICCA 2004), Myanmar 8th January 2004.

[12]Hung Q. Ngo, Anjum Shehzad, Saad Liaquat, Maria Riaz and Sungyoung Lee. Developing Context-Aware Ubiquitous Computing Systems with a Unified Middleware Framework. Proceedings of Embedded and Ubiquitous Computing: International Conference EUC 2004, LNCS Volume 3207, Springer Verlag 2004, pp. 672 – 681.

[13] http://uclab.khu.ac.kr

[14] Wang Jin, Shu Lei, Jinsung Cho, Young-Koo Lee, Sungyoung Lee, Yonil Zhong. "A Load-balancing and Energy-aware Clustering Algorithm in Wireless Ad-hoc Networks". The 1st International Workshop on RFID and Ubiquitous Sensor Networks, USN'2005.
[15] Xiaoling Wu, Shu Lei, Wang Jin, Jinsung Cho, Sungyoung Lee, "Energy-Efficient Deployment of Mobile Sensor Networks by PSO." Advanced Web and Network Technologies, and Applications (APWeb 2006), Harbin, China, January 16-18, 2006, pp. 373-382.
[16] Wendi B. Heinzelman, Anantha P. Chandrakasan, and Hari Balakrishnan: An Application-Specific Protocol Architecture for Wireless Microsensor Networks. IEEE Transactions on Wireless Communications, Vol. 1, No. 4, pp. 660-670, 2002.

5.4 Conditional Query Aggregation Algorithm

5.4.1 Introduction

Sensor networks consist of a large number of nodes with sensing, computation and communication capabilities and are able to process massive amounts of data. Recently, one unified view is to treat them as distributed databases to provide data services [8], [9], [13], [15], [16], [17], where sensors are programmed via declarative queries in a variant of SQL or an event-based language. Queries are used for end users to request data from these databases. However, since most of the sensor nodes are limited in power, computational capacities and memory, how to devise query processing strategies that reduce the energy consumption in dissemination and transmission becomes more and more important for highly energy-constrained sensor networks.

Traditionally, end users submit queries to the base station where there is a query manager processing the queries one by one (local cache strategies may be used) and disseminating these queries to the proper regions of sensor networks via the underlying routing infrastructure. However, for some applications, multiple end users may issue a large number of simple queries simultaneously, which means that the query rate can be very high. Thus the overhead of query dissemination can no longer be ignored as usual, and the energy consumption spent on sending and routing queries and data of response may increase. In addition, a large number of query messages disseminated in networks may cause serious traffic congestion. In such cases, reducing queries to be disseminated can significantly reduce the energy consumption of redundant query propagation and data transmission in sensor networks. Thus the overall performance of sensor networks can be improved.

In our previous work [6], we proposed an effective query aggregation algorithm SAQA to reduce the overall energy overhead for the data services in the sensor networks. However, in [6], we constructed the aggregated regions based on the input query regions and did not consider the existing topology and distribution of sensors in the networks. Such as, adjacent spatial information can be combined together in that they may share part of the route for query dissemination and data transmission. In addition, how to decide the query merge order is an important issue we should consider.

In this chapter, a two-tier buffer model is proposed to conduct the query aggregation. In this model, a conditional query aggregation algorithm is provided to reduce the redundant information in the queries and the number of queries for dissemination. We concentrate on query aggregation. The sensor network performs in-network aggregation while routing data from source sensor nodes through intermediate sensor nodes to the base station. Specifically, the two major contributions of this chapter are: 1) a two-tier buffer model to conduct the query aggregation, and 2) an effective conditional query aggregation algorithm which consolidates queries based on spatial, attribute and deadline query information and network topology information. The expected benefits include savings in the overall energy consumption, both in query dissemination and in data transmission.

The remainder of this chapter is organized as follows. In Section 2, we review the state-of-art research on query processing in sensor networks. Section 3 introduces the query model and our two-tier buffer model used in this chapter. In Section 4, our conditional query aggregation algorithm is detailed. In Section 5, performance evaluation and analysis results are given. Finally we conclude our study with scope for future work in Section 6.

5.4.2 Related Work

In the context of query processing, a lot of existing work in the literature deals with data aggregation, such as, AODV [3] adopted and enhanced in Cougar [9], Directed Diffusion [4], ACQUIRE [7], TAG [8]. Not much attention has been paid to conduct query aggregation. In some applications as mentioned above, query aggregation plays an important role of reducing energy overhead.

In [10], the authors introduced a general framework for distributed processing of spatial-temporal queries in a sensor network that has two main phases: (1) routing the query to the spatial area specified in the query; (2) collecting and processing the information from the nodes relevant to the query. They proposed the spatial-temporal query processing algorithm based on the local storage. In [11], the authors gave the energy consumption model used in the algorithm in [10] and also proposed three algorithms: WinFlood, FullFlood and WinDepth. Samuel et al. discussed the design of an acquisitional query processor (ACQP) for data collection in sensor network in [12]. They provided a query processor-like interface to sensor networks and used acquisitional techniques to reduce power consumption. In the query dissemination optimization, they built a semantic routing tree (SRT). SRTs provide an efficient mechanism for disseminating queries and collecting query results for queries over constant attributes. A specific probabilistic model based on time-varying multivariate Gaussians was provided in [14]. The authors assumed that the queries sent by users include error tolerances and target confidence bounds that specify how much uncertainty the user is willing to tolerate. Moreover, their work was focused on multiple snap-shot queries over the current state of the network, rather than continuous queries. With these strict assumptions, their application scope was narrowed down.

Although, our two-tier buffer model is different from those described above, there are a number of common elements. Our model supports the high query rate applications whereas many of the above authors did not consider the number of queries and conduct any query aggregation to reduce redundancy in original queries before dissemination.

In [2], the authors proposed a multi-layered overlay-based framework consisting of a query manager and access points (nodes), where the former provides the query aggregation plan and the latter executes the plan. They present an effective query aggregation algorithm, which is mainly based on reducing the duplicate/overlapping spatial information of the original queries sent by the applications to minimize the number of queries sent out and prevent data transmission in the same region happening multiple times. This work is similar to our previous work in [6]. However, in [2], no attention has been paid to the attribute and deadline information in the original queries. In

addition, they did not consider about the network topology when reducing the duplicate/overlapping spatial information in the queries.

5.4.3 System Model

5.4.3.1 Network Model

A wireless sensor network (WSN) consists of base stations and a number of wireless sensor nodes. Each sensor node is capable of collecting various data from the environment. End users issue queries to the base stations, which disseminate these queries to the network. Different mechanisms have been proposed to solve these queries and route back the data of response. We have the following basic assumptions for sensor networks:

- Sensor nodes remain stationary at their initial location, which means the topology is fixed,
- sensor field is divided into different regions after the sensor nodes are deployed, and the end users have the information about this division,
- there is a centralized query manager at the base station and it executes the query aggregation plan with a global view of the network topology,
- each sensor is able to track one or multiple values of certain variables such as temperature, pressure, humidity, etc.

5.4.3.2 Query Model

End users request information from a sensor network through queries. According to the features of applications, various queries can be injected into the sensor network [8], [15], [16]. In this chapter, we define the following tuple as our query model:

$Q = \langle S, A, T, F, D \rangle$, where, $S = \{s_1, s_2 \ldots s_M\}$ denotes the spatial information that indicates the geographical regions of the network. Since the end users can be interested in more than one region, the definition of S is extended to be a set of M regions. A denotes the attribute information the end users are interested in, such as, temperature, light, voltage, pressure, humidity, etc. As mentioned in the assumptions above, the definition of A is also extended to be a set $\{a_1, a_2 \ldots a_K\}$ of K attributes. *T (optional)* denotes the temporal information, indicating the duration of the query in time units, such as, seconds. The duration can be notated either as a single constant or as the elapsed time between two specific times, i.e. $T = c$ or $T = (t1, t2)$ where the time $t2$ is later than the time $t1$. *F (optional)* denotes the frequency information, indicating the frequency at which the data should be reported. *D (optional)* denotes the deadline information, indicating the urgency at which the data should be reported. Intuitive details are provided by the following example.

A typical query could be: "*find the temperature level and humidity level in region* s_1, s_2 *from 8am to 4pm every ten minutes and report the data back before 6pm*", where S is $\{s_1,$

s_2}, *A* is *{temperature level, humidity level}*, *T* is *(8am, 4pm)*, *F* is *600 seconds* and *D* is *6pm*. If there is no deadline information, we define *D* to be infinite.

To conduct query aggregation, we have the following assumptions about the query model:

- the query manager at the base station translates the queries to the above format. In the process of translation, the query manager provides the syntax check and verification of applications, such as the query manager ensures the t2 < D constraint;
- most of the queries have spatial information. A query without spatial information can be processed through the traditional query processing techniques such as flooding or direct diffusion [4], etc.;
- *F* in the majority of queries is constant. In other words, irregular occurring frequencies are not discussed in this chapter.

Our application scenario, in particular, focuses on the application where multiple end users may issue a large number of simple queries simultaneously. Such application scenario is very useful for some urgency processing attracting lots of end users' attention. Although the query model proposed above apply to the general applications, for ease of presentation in this chapter, the aggregation is done mainly based on the spatial information, attribute information and deadline information, so that we restrict our query model to a simpler one: $Q = < S, A, D >$.

5.4.3.3 Two-tier Buffer Model

In our two-tier buffer model, we allow multiple end users to submit a large number of queries. As new queries are generated, they are not sent immediately to the network for evaluation, but are gathered by the query manager at base station. After receiving the queries from end users, the query manager at base station first translates the queries to the query model above. Then it looks up the data in the local database. If the related data is already available in the local cache, the queries are solved locally to avoid redundant query dissemination. Otherwise, the query manager begins to conduct the query aggregation. The purpose is to reduce the number of query messages for dissemination and also the transmission overhead of aggregated queries and responded data.

In terms of the query manager, many unprocessed queries are still physically existing at base station and waiting for the processing of query manager. Therefore, there should be a physical buffer in the query manager at base station to store the unprocessed queries. We regard this buffer as the **First Tier Buffer (FTB)**. The **Second Tier Buffer** (**STB**) is the buffer that logically exists inside query manager which enables the query aggregation. In this case, we propose the Two-Tier Buffer model for our query aggregation work as Figure 1 shows.

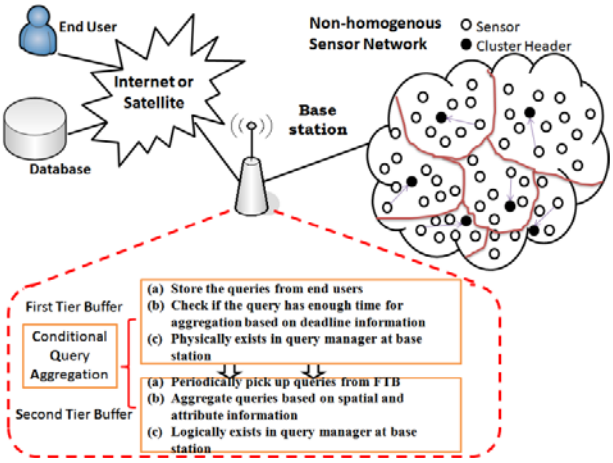


Fig.1 Two-tier Buffer Model

In FTB, queries are stored and deadline information is used to check whether the query has enough time for aggregation, in case that some queries may have urgent deadlines. If time is adequate for aggregation, the query is stored in FTB for later aggregation. Otherwise, no aggregation will be done for the query and it will be disseminated to the in-network immediately to meet the deadline requirement.

Queries are processed once every *epoch* (E_p) [18]. Each epoch consist of a *query aggregation* (*QA*) and a *query process* (*QP*) phase. In the *QA* phase, Query manager periodically collects all the queries stored in FTB and starts the aggregation in STB by our conditional query aggregation algorithm. More details about our conditional aggregation algorithm are presented in Section 4.2. In the *QP* phase, query manager disseminates the aggregated queries to the network and data of response are forwarded back to the query manager. The duration of a *QP* phase is a tunable application-specific parameter.

In fact, the buffer size is limited and cannot store all the queries sent by end users, especially when the query rate is very high. Storage strategies are needed. In this chapter, we adopt the FIFO approach, where the newly generated query is simply discarded if the buffer is full at that time or the end users could send the query again later.

5.4.4 Conditional Query Aggregation (CQA)

5.4.4.1 Problem Definition

According to our query model mentioned above, we assume there are N queries: Q_1, Q_2 ... Q_N. The main objective of query aggregation is to reduce the number of query messages for dissemination and thus reduce the overhead energy consumption in both query dissemination and data transmission. To conduct the query aggregation operation, we define our conditional aggregation function as: $f(Q_1, Q_2 \dots Q_N) = \forall_{(D_i > E_p)}\{Q_1, Q_2 \dots Q_N\} \rightarrow \{Q_1', Q_2' \dots Q_R'\}$, where D_i is the deadline

information in Q_i $(1 \le Q_i \le N)$. V denotes the aggregation operator. The energy consumption for f with output queries $\{Q_1', Q_2' \dots Q_R'\}$ $(1 \le Q_i' \le R)$ is defined by

$$E_f = \sum_{i=1}^{R} [\, G(Q_i') + H(Q_i') \,] \quad (1)$$

where $G(Q_i')$ represents the energy consumption for query dissemination from the query manager at base station to the proper sensors, $H(Q_i')$ represents the energy consumption for data transmission from source sensors through intermediate nodes back to the query manager at base station. Our objective is to find an optimal mapping so that the value of E_f is minimal. No query aggregation is just a special case of the conditional aggregation function, i.e., $N = R$ and $Q_i = Q_i'$, for all $1 \le i \le N$. In the next section, we provide the algorithm in detail.

5.4.4.2 Proposed Algorithm

Our conditional query aggregation algorithm is defined into the following steps:

Step 1: Initialization. Multiple end users send a large number of simple queries simultaneously to the query manager at base station. FIFO approach is adopted, where the newly generated query is simply discarded if the buffer is full at that time.

Step 2: In FTB, the query manager uses deadline information to check whether the query can be aggregated or not. A query that can be stored in FTB should satisfy the following criteria:

$$D > Ep \quad (2)$$

If the above inequality is satisfied, the query will be stored in FTB and then when a new epoch arrives, it is sent to STB for further aggregation. Otherwise, the query is treated as an urgent query and disseminated into the sensor network immediately.

Step 3: In STB, spatial and attribute information are used to aggregate queries. Query manager periodically picks up all the queries in FTB, consolidates the queries by our conditional aggregation algorithm and restores the recombined queries in terms of the spatial information in STB. After all the aggregation is done, query manager dispatches the aggregated queries via some location-based routing scheme.

The exhaustive approach to find the optimal mapping in our conditional aggregation function, in other words the optimal merge order, is obviously impractical for real applications especially when the number of query messages increases. To solve this problem, we consider the network topology when consolidating the spatial information. The basic idea is: As the query manager has a global view of all the query information and the network topology, it can calculate all the overlapping regions in all the queries in FTB, find an optimal merge order based on the most adjacent regions information. Queries containing most adjacent spatial information are merged together with higher priority. The reason is that routing to adjacent regions is probably sharing part of the dissemination paths. Thus, less intermediate nodes are involved in both the query dissemination and data transmission and energy consumption is consequently reduced.

For ease of presentation, we consider the following examples:

Suppose we have two queries $Q_1 =< (s_1, s_2, s_3), (a_1, a_2), D_1 >$ and $Q_2 =< (s_1), (a_1), D_2 >$ coming simultaneously from end users and the queried information is not available in base station local database, and both queries have enough time for aggregation ($D_1 > E_p$ and $D_2 > E_p$). In this case, query manager only needs to store Q_1, since the spatial and attribute information in Q_2 are completely included in Q_1..

Given another example, two queries Q_1 and $Q_3 =< (s_2, s_3, s_4), (a_2, a_3), D_3 >$ and we suppose $D_1 > D_3$. They ask for partially different attributes in overlapped regions, recombining them into three simple queries as $Q_1' =< (s_2, s_3), (a_1, a_2, a_3), D_3 >$, $Q_2' =< (s_1), (a_1, a_2), D_1 >$ and $Q_3' =< (s_4), (a_2, a_3), D_3 >$ will be much more beneficial. It reduces the energy overhead of sending separate duplicate query messages to the same region and more importantly avoids disturbing other intermediate nodes (and nodes in the overlapping region) while routing the queries. Different attributes can be collected at the same time in the same region to satisfy the original queries. Compared to the original case, the number of intermediate nodes (and nodes in the overlapping region) involved in the routing process is reduced. In addition, duplicated data transmission is also prevented.

The basic idea has some similarities to our previous work in [6]. However, in [6], we constructed the aggregation without considering the existing topology and distribution of sensors in the network. In this chapter, we take the network topology into account when selecting queries for aggregation according to the spatial information. For instance, we assume three queries Q_4, Q_5 and Q_6 query three different regions s_4, s_5 and s_6. Geographically, s_4 and s_5 are adjacent areas where s_6 is far away from both. Such adjacent spatial information can be used to conduct aggregation so that Q_4 and Q_5 may be recombined. Adjacent areas are most probably sharing part of the route for query dissemination and data transmission. In this case, sending Q_4 and Q_5 within one packet may reduce the energy overhead when routed the common path.

Before the detailed algorithm, we provide some definitions and the rule of adjacent region selection used for finding optimal merge order.

Definition I: If A is a set, let |A| represent the number of elements in A.

Definition II: Suppose we have two regions s_i and s_j. The nodes sets on the routes from base station to s_i and s_j are defined as N_{s_i} and N_{s_j}. If s_i and s_j are adjacent, they probably share some intermediate sensor nodes. Thus the intersection of their routing node sets should be at least not NULL. Therefore, we define the weight of adjacent regions to be:

$$w_{i,j} = \frac{|N_{s_i} \cap N_{s_j}|}{|N_{s_i} \cup N_{s_j}|} \ (\mathrm{i} \neq \mathrm{j}) \qquad (3)$$

Adjacent Region Selection Rule: our intention is to disturb intermediated nodes as few as possible to avoid duplicate query propagation when the queries are targeting at adjacent regions. For all the regions in the sensor networks, we calculate the weights for each pair and choose the maximum value for that pair.

$$w = \max_{i,j}(w_{i,j}) \tag{4}$$

Iteratively using the above formula, we get all the adjacent regions and store this information in the query manager. When the query manager aggregates the queries according to the spatial information, it can refer to the adjacent regions information to choose the adjacent queries to be merged.

In our network model, we assume the network topology is fixed. However, routing is independent of network topology. In other words, a routing algorithm can dynamically choose routing paths, making N_{s_i} and $w_{i,j}$ changed. To solve this problem, the query manager can periodically calculate $w_{i,j}$ for all the regions.

Through the Adjacent Region Selection Rule, we can get the set W with C adjacent regions. C is the number of adjacent regions in W.

Query manager periodically picks up all the queries in FTB, consolidates the queries by our conditional aggregation algorithm. For the query aggregation operation, we mainly use two important concepts, union and intersection, in set theory.

For two queries Q_i, $Q_j \in Q$, we say these two queries are overlapped, if

$$S_i \cap S_j \neq \emptyset \tag{5}$$

Referring to the adjacent regions set W obtained through the Adjacent Region Selection Rule, the query manager aggregates the overlapped queries:
CASE I: If $S_i \subseteq S_j$ (the same with $S_j \subseteq S_i$) as in Figure 2, after the aggregation, we will get:

$$S_{i,j} = S_i \cap S_j, A_{i,j} = A_i \cup A_j, S_{rj} = S_j - S_{i,j}, A_{rj} = A_j \tag{6}$$

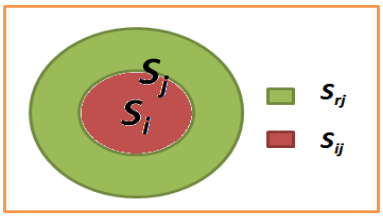


Fig.2 Two overlapped regions Si and Sj, where $S_i \subseteq S_j$

Since S_i is included in S_j, we can separate the area to two parts, one is the intersection part $S_{i,j}$, and the other is the rest part of S_j after separation, S_{rj}. As for the attribute information, in the case of $S_{i,j}$, we unify the two parts from Q_i and Q_j to get one set $A_{i,j}$.

That is, what we have introduced, the union operation. And for the rest part of S_j, S_{rj}, the attribute in the new aggregated queries will be the same as in the original query S_j.

Thus the output queries will be:

$$< S_{i,j}, A_{i,j}, min\{D_i, D_j\} > + < S_{rj}, A_{rj}, D_j > \quad (7)$$

Nodes in the intersection area will be disturbed only once, but still gather the necessary information.

CASE II: If $S_i \not\subseteq S_j$ and $S_j \not\subseteq S_i$ as in Figure 3, after the aggregation, we will get:

$$S_{i,j} = S_i \cap S_j, A_{i,j} = A_i \cup A_j, S_{ri} = S_i - S_{i,j}, A_{ri} = A_i, S_{rj} = S_j - S_{i,j}, A_{rj} = A_j \quad (8)$$

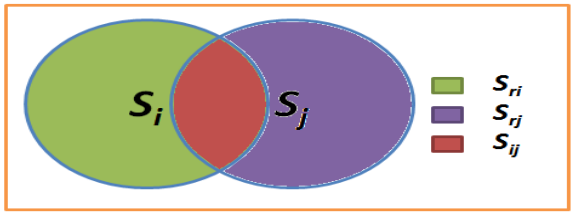


Fig.3 Two overlapped regions Si and Sj, where $S_i \not\subseteq S_j$ and $S_j \not\subseteq S_i$

As for the spatial information, Q_i and Q_j are partially overlapped with one intersection $S_{i,j}$. That's why we aggregate the two queries into three parts, one is the intersection, and the other two are the rest parts of Q_i and Q_j respectively. As for the attribute information, the same as in CASE I for $S_{i,j}$. For the rest two parts S_{ri} and S_{rj}, the attribute in the new aggregated queries will be the same as in the original queries. Finally, the output queries will be:

$$< S_{i,j}, A_{i,j}, min\{D_i, D_j\} > + < S_{ri}, A_{ri}, D_i > + < S_{rj}, A_{rj}, D_j > \quad (9)$$

5.4.5 Performance Evaluation

5.4.5.1 Experimental Model

We assume that each sensor work in free space mode with some experimental data introduced in [5]: the energy consumption of sending message is calculated by $E_{tx}(a, b) = E_{elect} * a + E_{fs} * a * b^2$ and the energy consumption of receiving a message is calculated by $E_{rx}(a, b) = E_{elect} * a$, where a is the message size and b is the message transmission distance between the sender and receiver, $E_{elect} = 50\ nJ/bit$, and $E_{fs} = 100\ pJ/bit*m^2$ ($1\ nJ = 1000\ pJ$ and $1\ MnJ = 1000\ nJ$). Since the energy consumed for processing queries and sensing data consists of only a very small portion of the overall energy consumption (node that energy consumed to process 100 million instructions almost equals that to transfer 10 bits of data), we do not take it into account in our calculation.

We define the A*djacent Region Degree (ARD)* as the ratio of *C/M*, where *C* is the number of adjacent areas decided by the Adjacent Region Selection Rule and *M* is the number of

areas predefined by the network topology. The range of *ARD* is [0, 0.5], where 0 represents the case that routing to every region uses different paths, no intersection exists in these routes and 0.5 represents the case that any two of the regions can be aggregated.

In our two-tier buffer model, we provide the conditional query aggregation algorithm to reduce the redundant information in the queries and the number of queries for dissemination. That is to say we mainly concentrate on query aggregation and the sensor network itself performs in-network aggregation while routing data from source sensor nodes through intermediate sensor nodes to the base station. In our related work, AODV[3], DD[4], ACQUIRE[7], TAG[8] and [10, 12, 14] are mainly focus on the in-network query processing, not much related to the query aggregation at base station. Thus we choose the following three query processing approaches to compare with our conditional query aggregation (CQA) algorithm:

1) Spatial based Query Aggregation (SQA)
In [2], the authors proposed a multi-layered overlay-based framework consisting of a query manager and access points (nodes), where the former provides the query aggregation plan and the latter executes the plan. They present an effective query aggregation algorithm, which is mainly based on reducing the duplicate/overlapping ***spatial*** information of the original queries sent by the applications to minimize the number of queries sent out and prevent data transmission in the same region happening multiple times. This algorithm is our first comparison.

2) Spatial and Attribute based Query Aggregation (SAQA)
In [6], a ***spatial*** and ***attribute*** based query aggregation algorithm SAQA is proposed. In SAQA, not only spatial information, but also attribute information is used to conduct query aggregation. However, the algorithm has three main limitations: a) the algorithm has one assumption that all the queries have enough time for aggregation. In some real-time applications, such assumption may not be satisfied; b) during the aggregation, each time only two queries are chosen for aggregation and no storage strategy is considered; c) the algorithm does not consider the network topology.
3) Conditional Query Aggregation without considering network topology (CQA-1)
To illustrate the function of two-tier buffer, we propose this approach similar to our CQA:

- In FTB: the same as in our conditional query aggregation algorithm. A physical buffer existing in query manager at the base station is used to store queries submitted by the end users. This approach also adopts the FIFO storage strategy. When there is no space in FTB, the query will be simply discarded or the end user could send it again later. In addition, the query manager also uses ***deadline*** information and criteria (2) to check whether the query has enough time for aggregation.
- In STB: similar to [6], query manager aggregates the queries from FTB based on the ***spatial*** and ***attribute*** information. However, no consideration about the network topology and distribution of sensor nodes in the network.

5.4.5.2 Performance Results

5.4.5.2.1 Comparison with SQA, SAQA and CQA-1

1) Energy Consumption on Number of Query Messages

Figure 4 shows the comparison of energy consumption in terms of the number of query messages. In this figure, X axis represents the total number of input queries from the end users and Y axis represents the total energy consumption. From this figure, we have the following observations:

- CQA and CQA-1 outperform both SQA and SAQA. For example, with a large number of queries, i.e., from 100 to 200, the energy consumption of SQA and SAQA can be 2~6 times more than that of CQA and CQA-1. The result matches our expectation in that with the two-tier buffer, CQA and CQA-1 can do aggregation on more queries stored in the buffer. The energy cost for both query transmission and data delivery has been significantly reduced.
- The performance of CQA-1 is similar to that of CQA in that both CQA-1 and CQA adopt the two-tier buffer model to allow more queries to be aggregated. Thus the energy consumption can be saved significantly. However, we can still find that CQA costs less energy than CQA-1, although the difference is not so much as that between CQA and SAQA. The reason is that CQA-1 does not consider the network topology when aggregating the spatial information, even though it also conducts aggregation based on spatial and attribute information and uses two-tier buffer to store the queries.
- SAQA performs better than SQA. The reason is that SAQA not only reduces the redundancy in spatial information but also in that of attribute information.
- The overall energy consumption is sensitive to the number of queries. The larger the number of query messages is, the more energy consumption. However, in CQA, this problem is alleviated because, when the number of queries increases, there will be more chances of overlaps between query regions and more chances of adjacent spatial information, which can be effectively utilized by our conditional query aggregation approach.

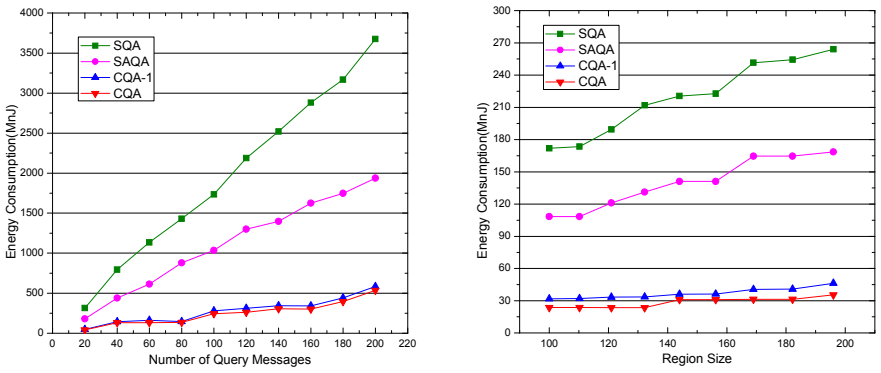


Fig.4 Energy Consumption on Number of Query Messages Fig. 5 Energy Consumption on Query Region Size

Although our initial motivation of this chapter is to reduce the overlapping information in the original queries and the number of query messages for dissemination when the query ratio is very high, our CQA shows competitive performance even in the case that query ratio is medium or low. In a word, our conditional query aggregation algorithm can efficiently process the queries and significantly reduce the energy consumption regardless of query ratio.

2) Energy Consumption on Query Region Size:

Figure 5 shows the comparison of energy consumption in terms of different query region sizes. In this figure, the X axis represents the different query region sizes and Y axis represents the total energy consumption. As the query size is enlarged, the overall energy consumption also increases. This is because a larger region size means that more sensor nodes are involved, or more query/data transmissions are performed. Regardless of the query size, our CQA performs better than the other 3 schemes with the same reasons given above.

5.4.5.2.2 Self-comparison with FTB size and ARD

1) Energy Consumption and Query Delivery Ratio on First Tier Buffer Size

Figure 6 shows the energy consumption and query delivery ratio on different first tier buffer size. Query Delivery Ratio is obtained by comparing total number of queries received by the query manager with total queries disseminated after aggregation.

With the increasing of first tier buffer size, the query delivery ratio is also increasing. This is explainable because with a larger buffer more queries can be stored and later aggregated by the query manager. If the buffer size is small and the number of query messages is large, the query manager will have to discard some of the queries in that the buffer is full.

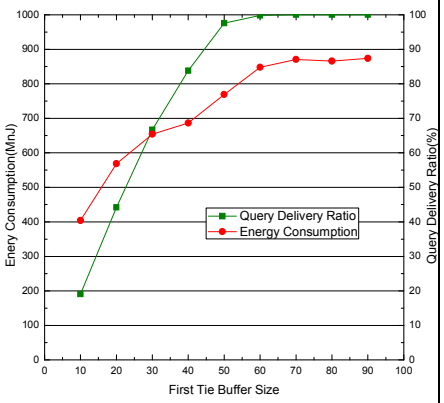


Fig. 6 Energy Consumption and Query Delivery Ratio on First Tier

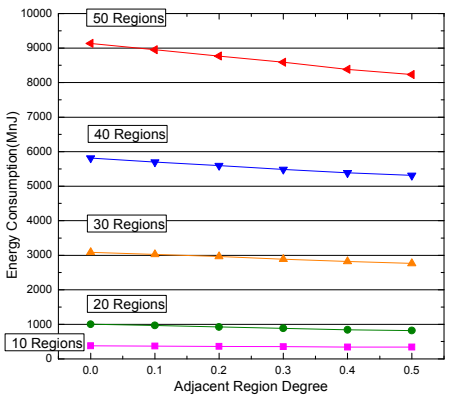


Fig. 7 Energy Consumption on Adjacent Region

Buffer Size	

In Figure 6, we can also find the variation of energy consumption according to the first tier buffer size. Basically, we have two observations: a) when the first tier buffer size is small, query delivery ratio is small. The smaller the query delivery ratio is, the more queries are discarded. In this case, only a small part of queries will be processed. Thus the energy consumption is small. b) However, when the buffer size increases, more queries can be processed, which means more queries disseminated after aggregation and more energy consumption for query dissemination and data transmission. Even though the energy consumption is increasing, the increasing ratio is not as fast as it is when the query delivery ratio is low. Because more queries stored in buffer also mean more reduction in spatial and attribute aggregation, which on the other hand decreases the energy consumption. A tradeoff between the first tier buffer size and energy consumption can be achieved with further research.

2) Energy Consumption on Adjacent Region Degree (ARD)

Figure 7 shows the energy consumption on different adjacent region degrees. In this figure, the X axis represents the different adjacent region degrees and Y axis represents the total energy consumption.

It is obvious in Figure 9 that the energy consumption decreases when the ARD increases. According to the definition of ARD, larger ARD implies that routing to adjacent regions shares more intermediate nodes. In that case, queries targeted at these adjacent regions can be combined together for dissemination. Less intermediate nodes will be disturbed for query dissemination and data transmission. Thus the energy can be saved. In addition, we have the observation that our CQA performs well especially with a large scale sensor network.

5.4.6 Conclusion and Future Work

In this chapter, we have proposed a two-tier buffer model to conduct query aggregation using an effective conditional query aggregation algorithm based on the spatial, attribute, deadline information in the original queries and also the information of network topology. The objective is to reduce overall energy consumption for query dissemination and data transmission in the sensor network. We have conducted extensive performance evaluations on different models. Our evaluation results show that by applying our two-tier buffer model and conditional query aggregation algorithm, we can significantly reduce the amount of query traffic and energy consumption for query processing regardless of query ratio.

There are still some directions to extend our study. First, we use two-tier buffer model to conduct query aggregation to save energy consumption at the price of storage. This proposes a tradeoff between the buffer size and reduction of energy. Second, if the number of queries in FTB is small, then STB may not be efficiently used. How to make full use of the second tier buffer to processing query aggregation needs further consideration. Finally, in our model, we only deal with fixed network, which means the

topology is stable. However, in most cases, the network topology can be changed. How to decide adjacent region in this changing network and how to make our conditional query aggregation algorithm adaptive are other issue for the future work.

5.4.7 References

[1] J. Beaver, M. A. Sharaf, A. Labrinidis, P. K. Chrysanthis, "Power-aware in-network query processing for sensor data", Proc. of Hellenic Data Management Symposium, 2003, 1 – 17.

[2] W. Yu, T. Le, Dong Xuan and W. Zhao, "Query Aggregation for Providing Efficient Data Services in Sensor Networks", Proc. of IEEE Mobile Sensor and Ad-hoc Sensor Systems (MASS), October 2004, 31-40.

[3] Ian D. Chakeres and Elizabeth M. Belding-Royer, "AODV Routing Protocol Implementation Design", Proc. of the International Workshop on Wireless Ad Hoc Networking (WWAN), Tokyo, Japan, March 2004, 698-703.

[4] C. Intanagonwisat, R. Govindan, D. Estrin, "Directed Diffusion: A scalable and robust communication paradigm for sensor networks", Proc. of the 6th Annual International Conference on Mobile Computing and Networking (MobileCom'00), Boston, Massachusetts, August 06-11, 2000, 56-67.

[5] H. O. Tan, and I. Korpeoglu, "Power efficient data gathering and aggregation in wireless senor networks", Proc. of ACM Special Interest Group on Management of Data (SIGMOD), San Diego, California, June, 2003, 66-71.

[6] J. Yang, B. Yan, S. Y. Lee, J. S. Cho, "SAQA: Spatial and Attribute based Query Aggregation in wireless sensor networks", The 2006 IFIP International Conference on Embedded & Ubiquitous computing (EUC'2006), Seoul, Korea, 1-4 August 2006, LNCS 4096, 15-24.

[7] N.Sadagopan, B.Krishamachari, A.Helmy, "Active query forwarding in sensor networks", Journal of Ad-hoc Networks, 3(1): 91-113, 2005

[8] Samuel Madden, Michael J. Franklin, Joseph M.Hellerstein, Wei Hong, "TAG: a Tiny AGgregation service for Ad-Hoc sensor networks", Proc. of the 5th symposium on Operating systems design and implementation, Boston, Massachusetts, December 09-11, 2002

[9] Y. Yao and J. Gehrke, "Query processing for sensor networks", Proc. of Conference in Innovative Data Systems Research, 2003

[10] Alexandru C, Mario AN, Jörg S, "A framework for spatio-temporal query processing over wireless sensor networks", Proc. of the 1st Int'l Workshop on Data Management for Sensor Networks in Conjunction with VLDB 2004. New York: ACM Press, 2004, 104−110.

[11] Alexandru C, Jörg S, Mario AN, "An analysis of spatio-temporal query processing in sensor networks", Proc. of the 1st IEEE Int'l Workshop on Networking Meets Databases in Cooperation with 21st IEEE Conf. on Data Engineering (ICDE 2005). Washington: IEEE Computer Society, 2005, 120−125.

[12] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong, "The design of an acquisitional query processor for sensor networks", Proc. of the 2003 ACM SIGMOD international conference on Management of data, San Diego, California, June 09-12, 2003, 491-502

[13] P. Bonnet, J. E. Gehrke, P. Seshadri, “Querying the physical world”, IEEE Personal Communications, Vol. 7, No. 5, October 2000, 10-15.
[14] Deshpande A, Guestrin C, Madden S, et al., “Model-Driven Data Acquisition in Sensor Networks”, Proc. of the 2004 International Conference on Very Large Data Bases, Toronto, Canada, May, 2004, 588-599
[15] Hellerstein, J., Hong, W., Madden, S., Stanek, K, “Beyond average: Towards sophisticated sensing with queries”, 2nd International Workshop on Information Processing in Sensor Networks (IPSN’03), March 2003, 63-79
[16] Madden, S., Szewczyk, R., Franklin, M., Culler, D, “Supporting aggregate queries over ad-hoc sensor networks”, Proc. of the Workshop on Mobile Computing and Systems Applications, Los Alamitos: IEEE Computer Press, 2002, 49-58
[17] C. Jaikaeo, C. Srisathapornphat, C. Shen, “Querying and tasking in sensor networks”, Proc. of the SPIE, Florida: SPIE, 2000, Vol.4037. 184-194
[18] Niki Trigoni, Yong Yao, Alan J. Demers, Johannes Gehrke, Rajmohan Rajaraman, “Multi-query optimization for sensor networks”, DCOSS 2005, 307-321

5.5 Fast Converging Pulse Coupled Oscillator Synchronicity Model in Sensor Networks

5.5.1 Introduction

Time synchronization is a critical piece of infrastructure in distributed system, and wireless sensor networks make particularly extensive use of synchronized time. Applications such as target tracking, MAC layer access schedule, time varying data collection, need either local or global synchronized time scale as a precondition.

Pulse Coupling Oscillator (PCO), as a model inspired from nature biological phenomena [1], has good scalability, strong robustness, self-organization and other salient features above the traditional synchronization protocols. It is suitable for large-scale sensor network outdoor observation applications. And the PCO provides a synchronicity primitive that achieves a tight alignment of individual nodes firing phases. It can be used in communication scheduling, coordinated duty cycling and time synchronization. [2]

All synchronization schemes, whether it is local or global, proactive or post-facto reactive [3], need some setup time and energy to get ‘chaos’ clock times synchronized. The length of the synchronizing time is called the immediacy of the time synchronization protocol. In some applications, like emergency detection (e.g., gas leak detection, intruder detection) [4], the immediacy requirement is very high. But usually there is a tradeoff between the immediacy and the energy efficiency. While our FPCO model proposed in this chapter can fast the synchronizing speed, meantime improve the energy efficiency by reducing almost half converging cycles and jump actions. The added overhead is negligible also.

The remainder of the chapter is organized as follow: section II introduces the related existing time synchronization protocols; section III introduces the classic mathematical model of PCO together with the node converging direction determinant formula. In section IV, given the inefficient action during the nodes converging process, we provide the improved fast converging PCO model. The proof of FPCO model is stated in section V. Section VI describes the simulation results and associated analysis. Section VII discusses the conclusion and future work.

5.5.2 Related Work

As the development of micro-electromechanical system (MEMS), producing large amount of low cost, small size sensor becomes possible. Wireless sensor networks attracted more and more research interest; consequently there appear many time synchronization protocols. According to the scalability, computing complexity and energy consumption characteristics, we classify current time synchronization protocols into three categories:

1) Centralized or partly centralized

All nodes' clocks are synchronized to some root node. These protocols mostly depend on specific network topology, like tree structure (TPSN [5] TDP [6], LTS [7]), hierarchical layer (FTSP [8]) or local star topology (RBS [3]).

TPSN (Time synchronization protocol in sensor network) uses pair-wise MAC layer timestamp messages propagate the root's clock along the spanning tree edge level by level. FTSP (flooding time synchronization protocol), differs with TPSN, nodes transmit and relay time information using flooding method, forming a dynamic hierarchical structure automatically. In receiver end it estimates four kind delays in transmission path and gets good time precision. RBS (reference broadcast synchronization), bases on the idea of receiver-receiver synchronization, removes the non-determinism caused by sending end completely. Although it is receivers' mutual synchronization, it also needs one central node broadcasting the pulse.

Centralized synchronization protocol has obvious shortcoming in large scale sensor networks. As the increase of intermediate node number, the precision will decrease. Meanwhile for the critical role of root node, centralized structure is easy to be attacked and destroyed. So although centralized protocol is simple and the precision is reasonable, its scalability and robustness is not very well.

2) Distributed synchronization protocols

These kinds of protocols are mostly used in sparse Ad Hoc networks. They don't synchronize the local clocks to some specific one node. Instead they generate time stamps according local clocks. Concretely time stamps are first transformed from local time to UTC ("Coordinated Universal Time", which is used as a common time transfer format) and then to local time of the receiver [9]. These transformations cannot be done exactly due to the unpredictability of the computer clocks, but will result in estimate lower and upper bounds. The TS/MS [10] protocol uses simple data collection algorithm to estimate the two neighbor clocks' relative drift and offset intervals then to get synchronized.

However, when it is used in the entire network, it also uses the hierarchical structure. RITS [11] (routing integrated time synchronization protocol) integrates post-facto time sync into a routing service. It performs inter-node time translation along the routing path.

Above the three distributed protocols' space and density scalability is much better than centralized protocols, however the time stamp conversion computing complexity is relatively high.

3) Synchronicity inspired from nature system:

In nature world, there is much large scale biological synchronicity phenomena, like populations of synchronously flashing fireflies, crickets that chirp in unison, electrical synchronous pacemaker cells.[] The earliest mathematic model on synchronization of pulse-coupled biological oscillators is Mirollo and Strogatz's work [12]. They proved that a very simple reactive node behavior would always converge to produce global synchronicity, irrespective of the number of nodes and starting times. After that many researchers give improvements by considering the non-ideal factors such as non-uniform coupled strength [13], local interaction [14], and transmission delay consideration [15-16]. On applications, besides the direct usage for time synchronization, PCO also can be used in data clustering [17] and data fusion field [18].

Biologically inspired time synchronization protocol can be either centralized or distributed. But here the PCO model is totally distributed. It maintains no internal state regarding neighbors or networks topology and just executes very simple computations and interactions. Second, self-organization: all the motions are inspired mutually by nodes themselves without any external command or request. Every node adjusts its own oscillate rhythms searching for a uniform pace. Third, robustness: it adapts to topology changes such as the loss and addition of nodes and links. Fifth, scalability: the more oscillators take part in the coupling procedure, the faster they will converge. The phenomena of thousands of fireflies' synchronous action also can prove PCO model's scalability is very good. Last one: low cost of resynchronization: as we know, the synchronized clocks will drift away as time goes by, we have to estimate the drift rate and compensate it [RBS] continuously or resynchronize them periodically [1, 4]. However, in PCO model, once some oscillators drift away a little, the whole group will adjust themselves to get resynchronization automatically by the pulse coupled between each other.

5.5.3 Mathematic model of PCO

5.5.3.1 Pulse coupled oscillator model

Today's computing devices are equipped with a hardware oscillator assisted clock. The PCO regards every distributed clock as an oscillator with the same fixed frequency but different initiate phase. "Pulse coupled" means using a fired pulse as signal to transmit oscillating information. The object is to let these oscillators act with the same phase, the same frequency, so-called synchronized. We use function $f(\phi)$ represent oscillator state, ϕ represents the oscillator phase. Phase ϕ and state $\varphi = f(\phi)$ both are defined on [0,

1]. The function cycle period is T. Here we require f is monotonically increasing and concave down (that is: $f^{'} > 0$ and $f^{''} < 0$; $f(0) = 0$, $f(1) = 1$). When there is only one oscillator, state will follow the function curve from 0 to 1 at a regular rate: $d\phi / dt = 1/T$. When it arrives to 1, it fires and emits a pulse, then resets phase to 0 and runs again. However, if during the walking path, at time t, it receives a pulse from the other oscillator, the state will jump an amount ε (Fig 5.1). The corresponding phase becomes $MIN(f^{-1}(f(\phi_t)+\varepsilon),1)$. And if the state jumps beyond 1, we set it as 1, and thinking it gets synchronized with the firing one. In such a way, oscillators interact with each other and adjust their phases to an agreed one.

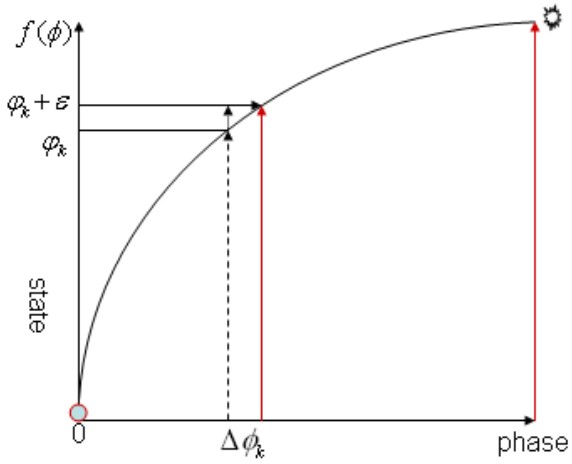


Figure 5.1 Oscillator jump action

5.5.3.2 Converging direction determinant formula of PCO

The discussion in [2] is extended and enhanced in this section to more clearly describe the dynamic nature of the converging process and get the converging direction determinant formula. The simple requirements on *f* ensure a node reacts more strongly to events that occur later in its time period. Here we give a graphic explanation on a pair nodes' converging procedure (Fig 5.2).

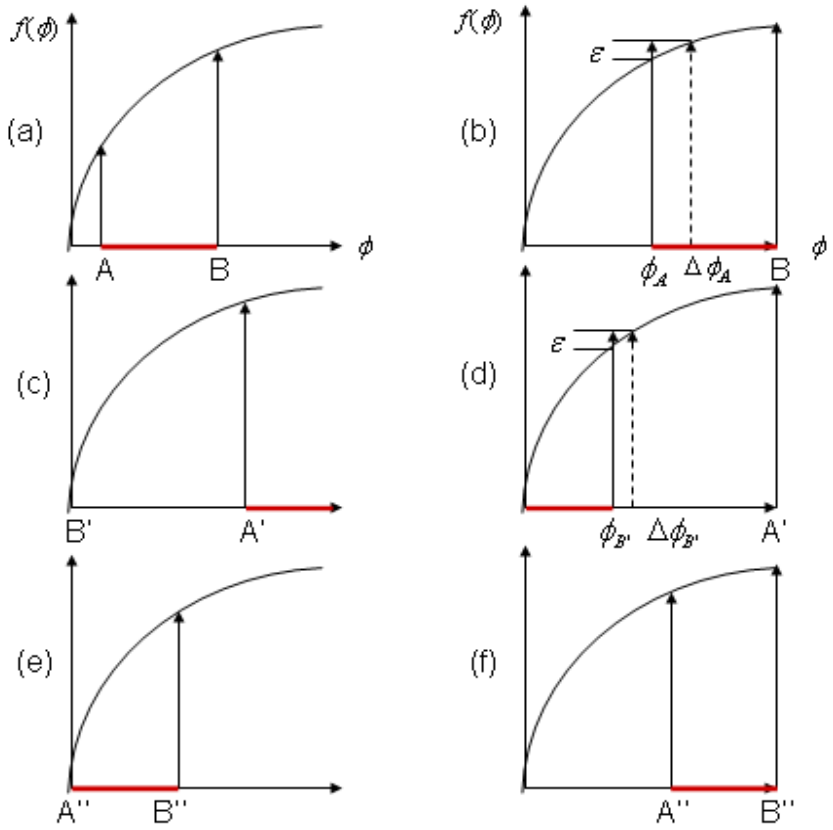


Figure 5.2 A pair nodes' phase converging procedure in one cycle. The red lines represent their "phase difference"

From observing two nodes' phase difference changing process in one cycle, we can infer the phase difference converge direction.

First introduce the symbols used in the deduction: $\varphi = f(\phi)$ is the oscillator function, φ represents the state; $\phi = f^{-1}(\varphi)$ is the inverse function of f . Here we use A, B represent two oscillators (or two already intra synchronized groups, which have to finish the final two groups synchrony stage) with different initial phases (we assume $\phi_A < \phi_B$) . Let $|\overrightarrow{AB}|$, the red (dark) lines in Fig 5.2, represent the phase distance vector from A to B. Whether $|\overrightarrow{AB}|$ shrink to 0 or enlarge to 1, we all think they get synchronized.

When oscillator B is fired, A jumps, AB distance changes to

$$|\overrightarrow{A'B'}| = |\overrightarrow{AB}| - \Delta\phi_A \qquad (1)$$

Temporarily the distance becomes smaller: $|\overrightarrow{A'B'}| < |\overrightarrow{AB}|$ (Fig 5.2 (b), (c));
Next after oscillator B firing, A jumps (Fig 5.2 (d), (e)), the distance becomes:

$$|\overrightarrow{A''B''}| = |\overrightarrow{A'B'}| + \Delta\phi_{B'} = |\overrightarrow{AB}| - \Delta\phi_A + \Delta\phi_{B'} \quad (2)$$

Now if we want to compare the value of $|\overrightarrow{A''B''}|$ and $|\overrightarrow{AB}|$, we need to know the relationship of $\Delta\phi_A$ and $\Delta\phi_{B'}$.
Using the definition of inverse function (Fig 5.1):

$$\Delta\phi_k = f^{-1}(\varphi_k + \varepsilon) - f^{-1}(\varphi_k) \quad (3)$$

Now we need to judge the monotonicity of $\Delta\phi_k$, using inverse function derivative rule, the first scale differential coefficient of $\Delta\phi_k$ is:

$$\Delta\phi' = \frac{1}{f'(\varphi+\varepsilon)} - \frac{1}{f'(\varphi)} = \frac{f'(\varphi) - f'(\varphi+\varepsilon)}{f'(\varphi)f'(\varphi+\varepsilon)} \quad (4)$$

For $f'' < 0$, f' is monotonically decreasing and $f' > 0$, $\Delta\phi' > 0$. $\Delta\phi$ is monotonically increasing.

Return to two oscillators' jump position ϕ_A and $\phi_{B'}$ (Fig 5.2 (d))

$$\phi_{B'} = 1 - \phi_{A'} = 1 - f^{-1}(f(\phi_A) + \varepsilon) \quad (5)$$

$$\phi_A - \phi_{B'} = \phi_A + f^{-1}(f(\phi_A) + \varepsilon) - 1$$

So when ϕ_B is at critical fire point ($\phi_B = 1$).

If $\phi_A + f^{-1}(f(\phi_A) + \varepsilon) > 1$, $\phi_A > \phi_{B'}$, $\Delta\phi_A > \Delta\phi_{B'}$, $|\overrightarrow{A''B''}| < |\overrightarrow{AB}|$.
$|\overrightarrow{A''B''}| < |\overrightarrow{AB}|$. For we set $\phi_{B''} = \phi_B = 1$, $\phi_{A''}$ must be bigger than ϕ_A, so in the afterwards, it will always meet the requirement $\phi_A + f^{-1}(f(\phi_A) + \varepsilon) > 1$, $|\overrightarrow{AB}|$ value continues shrinking until equal to 0.

If $\phi_A + f^{-1}(f(\phi_A) + \varepsilon) < 1$, $\phi_A < \phi_{B'}$, $\Delta\phi_A < \Delta\phi_{B'}$, $|\overrightarrow{A''B''}| > |\overrightarrow{AB}|$.
For the same reason, new phase of A will always drop in this class, $|\overrightarrow{AB}|$ continues enlarging until equal to 1.

If $\phi_A + f^{-1}(f(\phi_A) + \varepsilon) = 1$, $\phi_A = \phi_{B'}$, $\Delta\phi_A = \Delta\phi_{B'}$, $|\overrightarrow{A''B''}| = |\overrightarrow{AB}|$. This is the unique fixed point. Actually, for the uniqueness of this point and clock drift or clock measure errors, once the phase drifts away a little, the system will repel the phase to 1 or 0 synchronicity state. So the existence of unique fix point will not cause system pause and it is a repeller as described in [12].

Define $V_k = \phi_k + f^{-1}(f(\phi_k) + \varepsilon)$ (k represents the initial phase of the second oscillator), then we give out pair oscillators' converging direction determinant formula:

If $V_K > 1$, the phase difference converges to 0;

if $V_K < 1$, the phase difference converges to 1.

From the format of the determinant formula, we can see the pair' initial phase states, the coupling parameter ε and the extent to which $f(\phi)$ is concave down all influence the converging procedure, so synchrony is a "cooperative effect between the coupling and the dissipation; convergence disappears when either the coupling or the dissipation is removed" [12].

5.5.4 Fast converging PCO

5.5.4.1 Swing actions of phase difference

From pair oscillators' converging procedure, we notice that the "phase difference" swings: $|\overrightarrow{AB}| > |\overrightarrow{A'B'}|$ *and* $|\overrightarrow{A'B'}| < |\overrightarrow{A''B''}|$. The distance decreases and increases again in one cycle. Except the jump before final synchronicity, every time the jump do not cause the distance change by $\Delta\phi$ but $\Delta(\Delta\phi)$. In Fig 5.3 we draw out the phase difference changing procedure of a pair of oscillators' convergence, the blue (upper) line describes the swing action. The unnecessary swing between two directions causes converging time longer and consumes more energy.

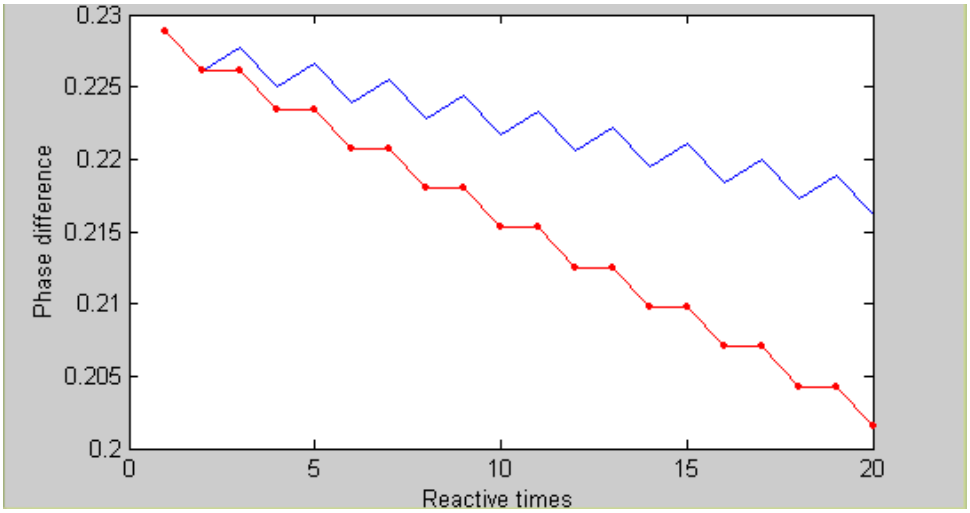


Figure 5.3 swing action of the phase difference

5.5.4.2 Single direction converge

The swing procedure in multi-oscillator situation is actually searching and attempting the final converging direction. When oscillators enter different synchrony groups, oscillators in one group always act simultaneously. Then we can regard this kind of group as one oscillator, the two synchrony groups can be thought as there are only two oscillators

mutually interacting. For pair of synchrony groups, base on our determinant formula, it is easy to know their final converging direction. So if we know the converging direction in advance by calculating the value of V_k, then we can avoid the unnecessary reverse direction jump, improve the total converging time and so as to save the energy consumption.

To be concrete, when an oscillator hears one pulse, it calculates current V_k value and checks the coming jump is to its final converging direction or not. If it is, then jump the oscillator, otherwise, ignore it. Thus the reverse jump is replaced by just staying its original place, and the up jump fold parts in blue (upper) line become the level parts in red (lower) line as showed in Fig 5.3. For getting rid of the useless reverse jump the converging time can be reduced much. Also because there are only two groups left, once decide the current firing pulse cause to the right jump direction, the next heard pulse would definitely cause jumping to the reverse direction and should be ignored automatically. Besides, because V_k value changes monotonically, once decided the converging direction, it will not change again. We only need judge the converging direction once in the whole procedure. And this will bring a little increase on the whole overhead but not a little enhancement on the synchronicity rate. We call it as fast converging pulse coupled oscillator model (FPCO). Furthermore, our FPCO still keeps the simplicity, self-organization, robustness of the original PCO model.

Pseudo-code of single direction converge

```
While (Not synchronized) do
{
Heard firing pulse
If (the first time heard the pulse)
/* first time, we need to judge the
 final converge direction */
{Calculate V = (\phi_A + f^{-1}(f(\phi_A) + \varepsilon)
If (V >1)
        Flag=1;
else
        Flag=0
}
Else
  {
    If (flag)
      {React to the fire pulse;
       Flag=! Flag;}
    else
      {Ignore the fire pulse;
       Flag=! Flag;}
   }
}
```

5.5.5 Theoretical analysis of FPCO

In this section we give out the theoretical analysis of FPCO model. Our analysis mainly follows that of Geoffrey Werner-Allen and Geetika Tewari [2].

Often, a transformation is completely described by its eigenvalues and eigenvectors. We use the eigenvalue and eigenvector of the dynamic matrix to analyze the two oscillators' phase difference changing trajectories. Without influencing the function property and final results here we do some simplification: 1) $f(\phi) = \ln(\phi)$, $g(\phi) = f^{-1}(\phi) = e^{\phi}$; 2) ϕ_A starts at any initial phase difference with ϕ_B, and ϕ_B starts just before firing. One cycle finishes at the time just before B's next firing. Refer to our determinant formula; two kinds of initial states produce two different dynamics.

THEOREM. Two oscillators A and B, governed by FPCO dynamics, will be driven to synchrony irrespective of their initial phases.
Proof:
First we assign $\phi_A < \phi_B$, $\phi_B = 1$.
For the case of $V_A > 1$ initial state, oscillator B's firing will cause oscillator A jump, but A's firing will be ignored by B. So if the initial state is $\overrightarrow{\phi_n} = (\phi_A, 1)$, after one cycle, the return map is $\overrightarrow{\phi_{n+1}} = (g(f(\phi_A) + \varepsilon), 1)$. Substitute the expression for function $f(\phi)$ and $g(\phi)$ and algebraic simplification yields:
$\overrightarrow{\phi_{n+1}} = (e^{\varepsilon}\phi_A, 1)$;
$\overrightarrow{\phi_{n+1}} = M\overrightarrow{\phi_n}$
Here n denotes the cycle number. The matrix M is defined as

$$M = \begin{bmatrix} e^{\varepsilon}, 0 \\ 0, 1 \end{bmatrix}$$

Hence the algorithm can be described as a linear dynamical system in $\vec{\phi}$, where $\vec{\phi} \in [0,1] \times [0,1]$. M already is a diagonal matrix containing the eigenvalues of M : $\lambda_1 = e^{\varepsilon}$ and $\lambda_2 = 1$; the corresponding eigenvectors are $\overrightarrow{v_1} = (1,0)$, $\overrightarrow{v_2} = (0,1)$. The evolution of the system is most simply described in term of $Base = \{\overrightarrow{v_1}, \overrightarrow{v_2}\}$. Figure 5.4 illustrates the system's evolution along the directions of $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$. Because the initial ϕ_B is 1 and $\lambda_2 = 1$, so in the $\overrightarrow{v_2}$ direction ϕ_B will never change and always keeps 1. Trajectories only move on $\phi_B = 1$ level line. And for $\lambda_1 = e^{\varepsilon} > 1$, the trajectories will

approach to the $\phi_A = 1$ axis along the vector $\vec{v_1} = (1,0)$. Finally it will stop in the state of synchrony $\vec{\phi} = (1,1)$.

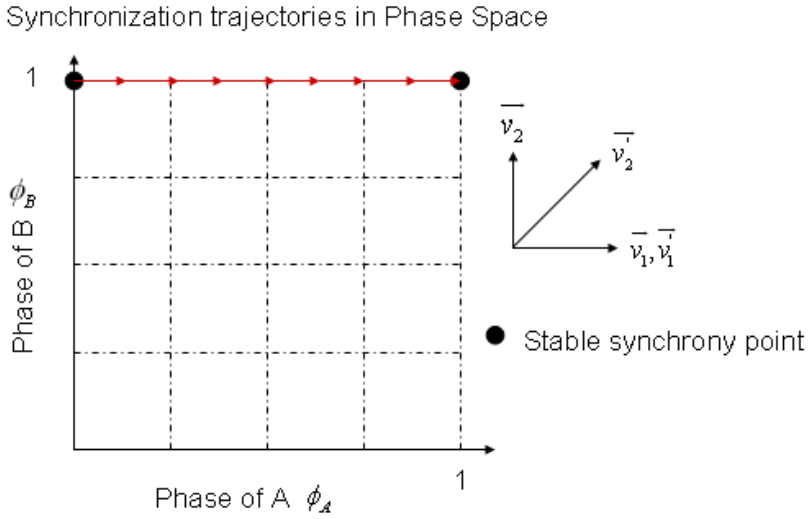


Fig 5.4 Trajectory of the oscillator phases

Second situation:

When the initial state meets $V_A < 1$, oscillator A will not respond for the firing of oscillator B, only B jumps after hearing the firing of A. So the return map changes to $\overrightarrow{\phi_{n+1}} = (1 - g(f(1-\phi_A) + \varepsilon), 1)$, $\overrightarrow{\phi_{n+1}} = (1 - e^{\varepsilon} + e^{\varepsilon}\phi_A, 1)$, still $\overrightarrow{\phi_{n+1}} = M\overrightarrow{\phi_n}$.

Here $M = \begin{bmatrix} e^{\varepsilon}, 1-e^{\varepsilon} \\ 0, 1 \end{bmatrix}$, eigenvalues and corresponding eigenvectors are $\lambda_1' = e^{\varepsilon}, \overrightarrow{v_1}' = (1,0)$; $\lambda_2' = 1, \overrightarrow{v_2}' = (1,1)$. Vector $\overrightarrow{v_2}'$ changes direction, but for $\lambda_2' = 1$, there will no movement along $\overrightarrow{v_2}'$ direction, it always stays on $\phi_B = 1$ level line. $\lambda_1' = e^{\varepsilon} > 1$, it's the same with the first situation, trajectories approach $\phi_A = 1$ axis along the vector $\overrightarrow{v_1}' = (1,0)$. Both of these two situations are illustrated in Fig 5.4.

5.5.6 Simulation result and discussion

Our experiment is set up in the ideal coupling environment: uniform interactions; all-to-all connected networks, no path loss; all kinds of processing, transmission, propagation delay are not under consideration. We choose $f(\phi) = \frac{1}{b}\ln[(e^b - 1)\phi + 1]$ as the oscillator function in our experiment. For the jump parameter ε and the extent of function b have similar effect on the rate of synchrony, we set b=1 and just change the value of ε to observe the synchrony rate of FPCO. All the comparison is under the same environment, initial state and parameter settings.

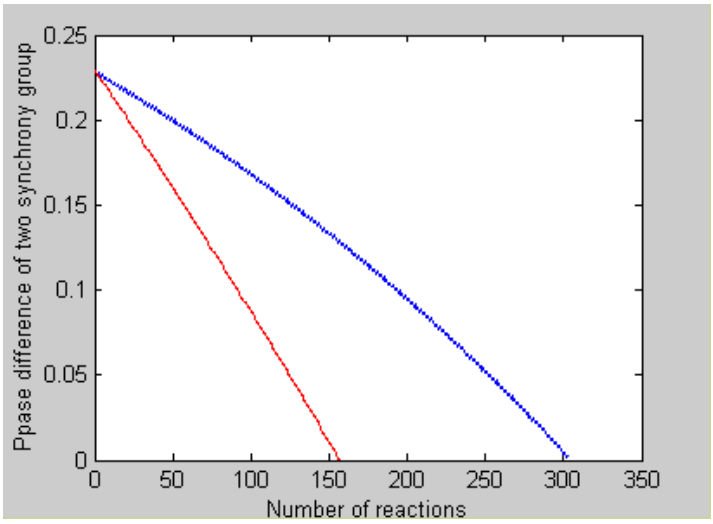


Fig 5.4 Two oscillators' synchrony procedure

Fig 5.4 shows one example of two oscillators synchrony procedure comparison between PCO and FPCO. Oscillator function is $f(\phi) = \ln[(e-1)\phi + 1]$, ε chosen as 0.002, and the initial phase is simulated by random numbers uniformly distributed in [0, 1]. Obviously for getting rid of the half of the attempting reverse jump actions. FPCO save almost half of converging time in this two synchrony group stage.

Table 1 is the comparison with different jump parameter ε (from 0.002 to 0.2). In order to decrease the influence of initial states, we repeat the test 5000 times with different initial states. Then get the average value for every ε. We can see that even under different ε values, our scheme always can save almost average 61.21% converging time in two synchrony group stage.

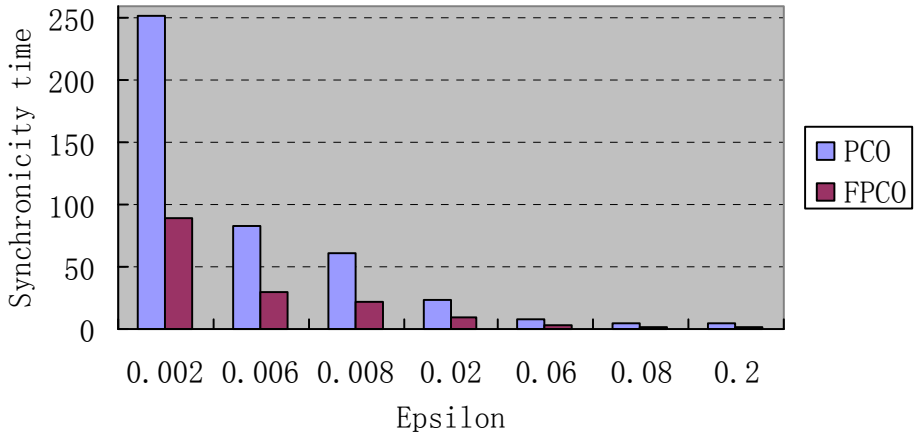


Table 1 Converging time comparison of two synchrony group stage

If we use fast PCO in the last two group stage, the whole synchrony time also will reduce. Table 2 (a), (b) give out experimental figures of improved converging speed in the whole synchrony procedure. The saved amount depends on the proportion of two synchrony group stage takes in the whole procedure. From our experiment results the averaged time saving is approximately 30%.

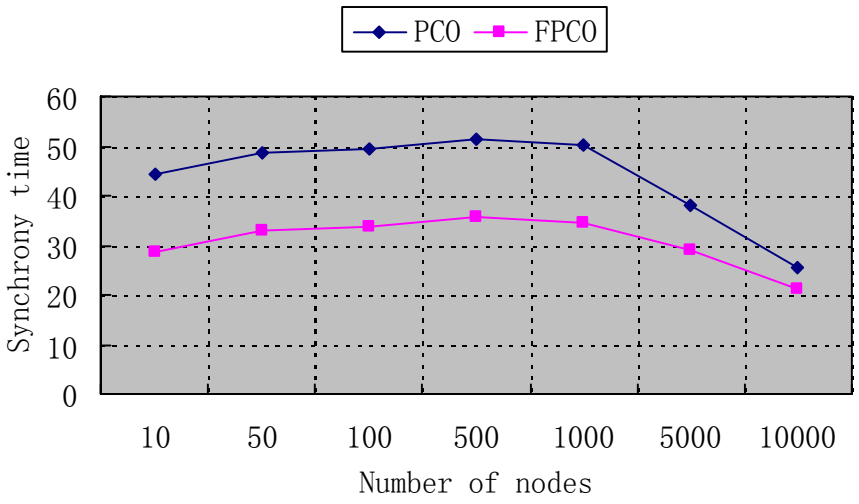


Table 2. Synchronicity time comparison of PCO/FPCO (b)

For the inherent scalability of PCO model, in the first column of Table2 (a), along with the increase of node number, the converging time increase not much, except some special initial state (like the node 500, 1000 situation).

Furthermore, the converging cycle is reduced by half; in every cycle, half of the jump actions are ignored. The total energy consumption on synchrony stage will be definitely saved.

5.5.7 Future Direction

Although our scheme only is used in the final stage of the converging procedure, it gets good result. We can see there still are spaces to use it in the whole synchrony procedure. That is from the two oscillators' situation extending to the n oscillators situation. But in the multi-oscillators situation, one oscillator will react more than one times in one cycle for hearing many pulses. It becomes not easy to decide the oscillators' final converging direction. Hence we need to further analysis the converging rules and modify our FPCO.

Our all analysis, experiments are under the ideal environment. If we want to adopt the FPCO in real sensor networks, the realistic effects in wireless environment must be considered and tested in real testbed. Further in real testbed our energy consumption saving estimation also can be verified and observed more exactly.

5.5.8 References

[1] S. H. Strogatz and I. Stewart, “Coupled oscillators and biological synchronization,” Scientific American Magazine, pp. 102–108, Dec 1993.
[2] G. W. Allen, G. Tewari, A. Patel, and R. Nagpal, “Firefly inspired sensor network synchronicity with realistic radio effects,” in SenSys05, San Diego, California, USA., November 2005.
[3] J. Elson, L. Girod, and D. Estrin, “Fine-grained network time synchronization using reference broadcasts,” in Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002), Usenix, 2002, pp. 147–163.
[4] Y. B. Sivrikaya Fikret, “Time synchronization in sensor networks: a survey,” IEEE Network magazine’s special issue on Ad-Hoc Networking, vol. 18, no. 4, pp. 102–108, 2004.
[5] S. Ganeriwal, R. Kumar, and M. Srivastava, “Timing-sync protocol for sensor networks,” in Proc. First Int. Conf. on Embedded Networked Sensor Systems(SenSys), Los Angeles, California, Nov 2003, pp. 138–149.
[6] W. Su and I. Akyildiz, “Time-diffusion synchronization protocols for sensor networks,” IEEE/ACM Transactions on Networking, vol. 13, no. 2, pp. 384–397, 2005.
[7] J. van Greunen and J. Rabaey, “Lightweight time synchronizationfor sensor networks,” in Proc. 2nd ACM Int. Workshop on Wireless Sensor Networks and Applications (WSNA 03), San Diego, California, Sept 2003, pp. 11–19.
[8] B. Kusy and M. Maroti, “Flooding time synchronization in wireless sensor networks,” in Proc. 2nd ACM. Conf. on Embedded Networked Sensor Systems(SenSys), Baltimore,Mariland, 2004, pp. 39–49.
[9] K. Romer, “Time synchronization in ad hoc networks,” in Proc. ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 01), Oct 2001, pp. 173–182.
[10] M. L. Sichitiu and C. Veerarittiphan, “Accurate time synchronization for wireless sensor networks,” in Proc. IEEE Wireless Communications and Networking Conference (WCNC 2003), New Orleans,LA, Oct 2003, pp. 1266–1273.
[11] J.Sallai, B. Kusy, A. Prabal, and P. Dutta, “On the scalability of routing-integrated time synchronization,” in 3rd European Workshop on Wireless Sensor Networks (EWSN), Zrich Switzerland, Feb 2006.
[12] R. Mirollo and S. Strogatz, “Synchronization of pulse-coupled biological oscillators,” SIAM., vol. 50, no. 6, pp. 1645–1662, 1990.
[13] L. F. Abbott, “A network of oscillators,” J. Phys. A, Math. Gen, vol. 23, pp. 3835–3859, 1990.
[14] D. Lucarelli and I. Wang, “Decentralized synchronization protocols with nearest neighbor communication,” in Proc. ACM SenSys 04, Nov 2004.
[15] U. Ernst, K. Pawelzik, and T. Geisel., “Delay-induced multistable synchronization of biological oscillators,” Physical Review E, vol. 57, no. 2, pp. 2150–2162, Feb 1998.
[16] W. Gerstner., “Rapid phase locking in systems of pulse-coupled oscillators with delays,” Physical Review Lett, vol. 76, no. 10, pp. 1755–1758, Mar 1996.
[17] Rhouma and H. M.B.H. andFrigui, “Self-organization of pulsecoupled oscillators with application to clustering,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 2, pp. 180–195, 2001.
[18] N. Wakamiya and M. Murata, “Scalable and robust scheme for data fusion in sensor networks,” in the First International Workshop on Biologically Inspired Approaches to

Advanced Information Technology (Bio-ADIT 2004). Lausanne: LNCS3141, Jan 2004, pp. 412–427.

5.6 Integrated Sleep-scheduling and Routing

5.6.1 Introduction

Ubiquitous sensor networks are expected to be widely employed in various applications such as medical care, military, environmental monitoring and industry [1]. Currently, energy supply is one of the fundamental bottlenecks. It is very costly and unpractical to replace sensor node batteries once they are deployed, both because of the large number of sensing nodes and the typically hazardous or unfriendly environment in which these nodes are deployed. Hence, prolonging network life becomes a primary concern in network design.

The sleeping technique has been used to conserve energy of battery powered sensors. Several researchers even suggest putting redundant sensor nodes into the network and allowing the extra sensors to sleep to extend network lifetime [2]. This approach is practical due to the low cost of individual sensors. When a sensor node is put into the sleep state, it is completely shut down, leaving only one extremely low power timer on to wake itself up at a later time. In a dense sensor network, rotating active and inactive sensors among the low power sensor members, some of which provide redundant data, is an intelligent way to manage sensors to extend its network lifetime. This leads to the following sleep scheduling problem: How does the cluster head (CH) or base station (BS) select which sensor nodes to be put into sleep, without compromising the sensing coverage capabilities of the whole network?

For a multi-hop USN rather than a single hop USN, an energy efficient routing protocol also needs to be considered. Hence, there exist more challenges than single hop networks, for example,

- The routing path (link) failure may happen during data transmission because of collision, node dying out (no battery), node busy, or other accidents. Some applications require real time information and data, which means retransmission is not possible. This motivates us to design a multi-path routing scheme for USNs.
- There exists energy constraint in USNs because most sensors are battery operated. This motivates us to consider energy aware routing.

In this chapter, we investigate the energy constraint problem in USNs and propose a sleep scheduling scheme in a single hop network based on Analytical Hierarchy Process (AHP). In addition, an integrated sleep scheduling and routing algorithm in a multi hop

environment is proposed again based on AHP. In a single hop network, three factors contributing to the optimal nodes scheduling decision are considered and they are 1) distance to CH, 2) residual energy, and 3) sensing coverage overlapping, respectively. Our goal is to balance energy consumption in low power sensor nodes and extend the sensor network lifetime while maintaining adequate sensing coverage capabilities. In a multi hop network, we propose an integrated AHP based sleep scheduling and multipath routing scheme for USNs, each of which has three different criteria considered as well. We evaluate the efficiency of both proposed schemes in terms of energy consumption, lifetime and coverage, and compare with related work, that is, random scheduling in heterogeneous sensor networks in the single hop case and Geographical Multipath Routing (GMR) scheme in the multi hop case.

The rest of the chapter is organized as follows. We introduce the background of AHP and related work in Section 2. The third section presents our sleep scheduling scheme in a single hop network. We present an integrated sleep scheduling and routing protocol in a multi-hop network in Section 4. Section 5 evaluates and analyzes the performance of the proposed method. Finally, we draw the conclusion and discuss future work in section 6.

5.6.2. Background

5.6.2.1 Analytical Hierarchy Process

The Analytical Hierarchy Process (AHP) [3] is a multiple criteria decision-making method which decomposes a complex problem into a hierarchy of simpler and more manageable sub problems. These sub-problems are usually called decision factors and weighted according to their relative dominances to the problem. AHP synthesizes their importance to the problem, and finds the best solution.

AHP performs following four main steps: decomposition, pair-wise comparison, local weight calculation, and weight synthesis.

A. Structuring Hierarchy

Structuring a problem as a hierarchy of multiple criteria is the first step of implementing AHP. The decision factors of the problem are identified and inserted into the hierarchy. The overall objective is placed at the topmost level of the hierarchy. The subsequent level presents the decision factors. The solution alternatives are located at the bottom level.

B. Calculating Local Weights

The second step is the evaluation stage where each factor is compared to all other factors within the same parent. Local weights consist of two parts: the weight of each decision factor to the goal and the weight of each nominee to each factor. Both of them are calculated with the same procedure. Taking the former as an example, we describe the procedure as the following three steps.

1) Making Pairwise Comparison

The evaluation matrices are built up through pairwise comparing each decision factor under the topmost goal. The comparison results are based upon user expertise experience

by asking questions such as "Which is more important and by how much?" These initial values are captured in square matrix A as

$$A = \left(a_{ij}\right)_{n\times n} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \quad (\mathbf{6})$$

where a_{ij} denotes the ratio of the i^{th} factor weight to the j^{th} factor weight, and n is the number of factors. The fundamental 1 to 9 scale can be used to rank the judgments as shown in Table 1 [20].

Table 1 A fundamental scale of 1 to 9

Number Rating	Verbal Judgment of Preferences
1	Equally
3	Moderately
5	Strongly
7	Very
9	Extremely

2, 4, 6, 8 indicate the medium value of above pairwise comparison.

The smaller one in a pair is chosen as a unit and the larger one is estimated as a multiple of that unit and assigned a number based on the perceived intensity of importance. Similarly, the reciprocals of these numbers are used to show the inverted comparison results. We thus obtain a reciprocal matrix where the entries are symmetric with respect to the diagonal.

2) Calculating Weight Vector

For the given matrix A in Eq. (1), its eigenvalue equation is written as AW = $\lambda_{\max}$W, where W is a non-zero vector called eigenvector, and $\lambda_{\max}$ is a scalar called eigenvalue. W and $\lambda_{\max}$ appear as a pair and cannot be taken apart. After standardizing the eigenvector W, we regard the vector element of W as the local weight of each decision factor approximately, which can be denoted as:

$$\mathbf{w}_j^T = \{w_1, w_2, \cdots, w_n\} \quad (\mathbf{7})$$

As a result, the weights of the decision factors can be achieved by calculating the eigenvector of AHP matrix and the eigenvalue that approximately equals the number of assessed elements.

3) Checking for Consistency

If every element in Eq. (1) satisfies the equations $a_{ij} = 1/a_{ji}$ and $a_{ik} \cdot a_{kj} = a_{ij}$, the matrix A is the consistency matrix. However, the evaluation matrices are often not perfectly consistent due to people's random judgments. These judgment errors can be detected by a consistency ratio (CR), which is defined as the ratio of consistency index (CI) to random index (RI). CI can be achieved by

$$CI = (\lambda_{\max} - n)/(n-1), \quad (\mathbf{8})$$

where $\lambda_{\max}$ is the eigenvalue and

$$\lambda_{\max} = (1/n)\sum_{i=1}^{n}(AW)_i / W_i \ . \quad (9)$$

The *RI* is given in Table 2 [3]. When $CR \leq 0.1$, the judgment errors are tolerable and the weight coefficients of the global weight matrix W_j are the weights of decision factor under the topmost goal. Otherwise, the pairwise comparisons should be adjusted until matrix A satisfies the consistency check, i.e. matrix A needs to be reinitialized.

Table 2: Random index

n	***RI***
1	0
2	0
3	0.58
4	0.90
5	1.12
6	1.24
7	1.32
8	1.41
9	1.45
10	1.49
11	1.51

C. Calculating Global Weights

From above steps, we can obtain not merely the weights of decision factors towards the topmost goal from W_j but also the weights of alternatives towards each factor. If there are k candidates, all the k weight matrixes of alternatives under n factors construct a $k \times n$ matrix, denoted as $W_{n_i/j}$, i=1, 2, … k, j=1, 2, …n.

The global weight of each alternative can be achieved through multiplying the local weight by its corresponding parent. So the final weight matrix in the symbol of W_{n_i} is calculated as

$$W_{n_i} = W_{n_i/j} \cdot W_j \ , \quad (10)$$

where the final weight of each alternative is calculated as

$$W_{n_i} = \sum_{j=1}^{n} W_{n_i/j} \cdot W_j \ . \quad (11)$$

The larger the final weight of alternative, the higher the probability it is eligible to be selected.

5.6.2.2 Related work

Sleep scheduling which aims to conserve the energy of the sensor nodes has been studied in the literature. In [2], nodes are allowed to sleep based on routing information, and nodes switch between sleep and active state based on the traffic of the network. As a modification to this basic algorithm, reactive features have been added to the node's schedule. The node would wake up more frequently based on the route discovery interval. Another widespread option is to turn off redundant nodes in the network [4]. In this scheme, the density of low power sensors is high enough to maintain the sensing coverage of the entire network even when some nodes are turned off. Each node studies the activity of their neighbors and decides to sleep if the coverage can be maintained by the active nodes. A back-off based approach has been used to prevent neighboring nodes to turn themselves off simultaneously.

In [5], a few nodes are selected as coordinators which would then decide the sleep/awake schedule of the other nodes in the network. While coordinators are awake at all times, the other nodes in the network sleep in order to conserve the overall network energy. In [6] nodes are randomly selected to go to the sleep mode and in [7] a Linear Distance-based Scheduling (LDS) technique has been used to define the sleep schedule of the nodes in a cluster based homogeneous network. In [8], the authors release the single hop communication assumption of [7] and introduce a Hop-based Sleeping Scheduling (HSS) algorithm in a circular sensor network which is divided by a number of levels. The overall result of these sleep schedules is a considerable reduction in the energy consumption of the sensor network.

In [9], the authors propose a cross-layer sleep-scheduling-based organizational approach, called SS-Trees, in order to increase monitoring coverage and operational lifetime of mesh-based USNs. An integer linear programming (ILP) formulation and an iterative algorithmic approach are suggested to determine the feasible SS-Tree structures for these purposes. The ILP approach requires the determination of objective functions and several constraints, which is often complicated. Hence the proposed AHP based approach is different and simpler in that we only need to give the estimated weight to several factors as an input for AHP to finish the whole process of optimal decision making without knowing the objective functions and constraints. From this point-of-view, AHP is easier to carry out with the achievement of the same performance goals.

Many routing protocols have also been developed for ad hoc networks, which can be summarized into two categories: table-driven (e.g., destination sequenced distance vector [10], cluster switch gateway routing [11]) and source-initiated on-demand (e.g., ad hoc on-demand distance vector routing [12], dynamic source routing (DSR) [13]). In [14], Lee and Gerla propose a Split Multi-path Routing protocol that builds maximal disjoint paths, where data traffic is distributed in two roots per session to avoid congestion and to use network resources efficiently. A Multi-path Source Routing (MSR) scheme is proposed in [15], which is an extension of DSR. Their work focuses on distributing load

adaptively among several paths. Nasipuri and Das [16] present the On-Demand Multi-path Routing scheme which is also an extension of DSR. In their scheme, alternative routes are maintained, which can be utilized when the primary one fails.

In sensor networks, location is often more important than a specific node ID. For example, in sensor networks for target tracking, the target location is much more important than the ID of reporting node. Therefore, some location-aware routing schemes have been proposed for USNs. A greedy geographic forwarding with limited flooding to circumvent the voids inside the network is proposed in [17], and some properties of greedy geographic routing algorithms are studied in [18]. Jain et al [19] proposes a geographical routing using partial information for USNs.

The AHP algorithm for CH selection is proposed in [20]. However, our approach is different from theirs in that we apply AHP to sleep scheduling and moreover use different AHP factor namely overlapping coverage.

5.6.3 Sleep-Scheduling in a Single-hop Cluster based Network

We adopt the same radio model as stated in [21] with ε_{fs} =10*pJ/bit/m*^{2} as amplifier constant, E_{elec}=50*nJ/bit* as the energy being dissipated to run the transmitter or receiver circuitry. It is assumed that the transmission between the common nodes or between the CH and its individual member node follows a second-order power loss model. The energy cost of transmission for sensor nodes at distance *d* from each other in transmitting an *l*-bit data is calculated as:

$$E_T(l,d) = lE_{elec} + l\varepsilon_{fs}d^2 \quad (12)$$

A typical sensor network could contain thousands of small sensors. In some specific applications, clustering has been employed to group a number of sensors, usually within a geographic neighborhood. In such a cluster based topology, sensors can be managed locally by a CH which is a node responsible for management in the cluster and for communication between the cluster and the base station.

We aim to enhance the efficiency of the given sensor network by enabling a balanced usage of energy across the nodes and an improved network lifetime without compromising network coverage. Figure 1 is the illustration of cluster based sensor network topology in which our proposed single hop sleep scheduling scheme is designed. We focus on energy consumption at the cluster level.

A. Assumptions

We consider the sleep node scheduling problem under several assumptions as follows:

- The target sensor network is heterogeneous with a large number of low power sensor nodes to serve as member nodes and a small number of more powerful nodes to serve as CHs. The motivation behind is to confine the complex hardware and additional battery to a few CH nodes. The low power nodes are simple in hardware and perform basic functions such as sensing and simple computations;

- A large number of sensor nodes are deployed over a sensing field, such that at least some sensor nodes can be put into the sleep state without degrading the sensing coverage of the network;

- The CHs can communicate directly with BS and vice- versa. Similarly, the CH can reach all the sensor members in the cluster in one hop and vice-versa. Thus, it is not needed for any routing strategy from the BS to any specific CH or from any CH to the individual sensor member.

- The application can tolerate some delay in reports from some sensors in each *round.*

B. Network Parameters

The user-defined parameters used in defining the network are listed below:

1) Fraction of nodes selected to sleep in a given round, '*r*': This is the fraction of the total number of nodes in the network that are selected to sleep in each *round*.

2) Threshold limit, 'θ': This denotes the fraction of nodes in the network, which, when dead, determines the lifetime of the network.

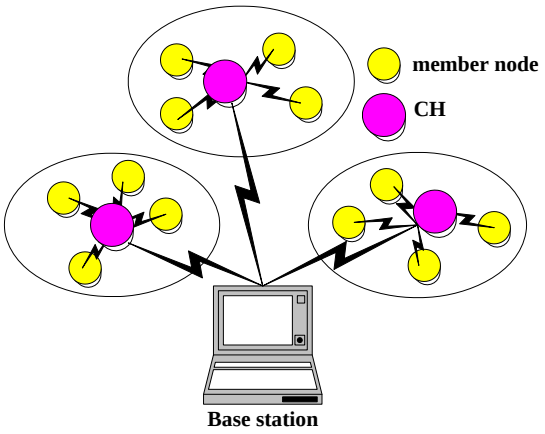


Fig. 1 Cluster based sensor network topology.

C. Sleep Scheduling Scheme by AHP

In our design, three factors influence the load balance and coverage directly, that is, 1) distance to CH, 2) residual energy, and 3) sensing coverage overlapping:

1) Distance to CH: Distance of a node to its CH. It can be approximated by the signal strength of radio transmission. The node with longest distance to the CH is preferred to be put into sleep.

2) Residual energy: Remaining battery of the sensor node. The initial energy is predefined. In addition, the energy consumption for transmission is calculated using Eq. (7) by CH.

3) Sensing coverage overlapping: Overlapped sensing range of a node by neighbor nodes. The node with larger overlapping degree, i.e., the node with higher redundancy, is desired to be selected as sleeping node.

This optimized sleep scheduling process is a multiple factors optimization problem and can be achieved by AHP, which is used to select the nodes eligible to sleep in one cluster. It is carried out in three steps:

Step 1: Collect information and formulate the sleeping nodes selection problem as a decision hierarchy of independent factors.

Step 2: Calculate the relative local weights of decision factors or alternatives of each level.

Step 3: Synthesize the above results to achieve the overall weight of each alternative nodes and choose the one with largest weight as the eligible sleeping node.

The goal of the decision "select a node eligible to sleep" is at the top level of the hierarchy as shown in Fig. 2. The next level consists of the three decision factors. At the bottom level there exist the *m* alternative sensor nodes to be evaluated.

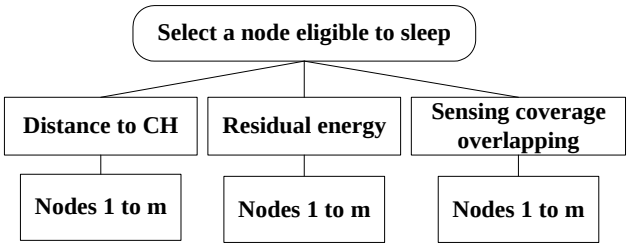


Fig. 2 AHP hierarchy for sleeping nodes selection in a single hop case.

In AHP modeling, the evaluation matrix A, here denoted as A1, is determined based on Eq. (1) as follows:

$$A1 = \begin{array}{c} \\ \alpha \\ \beta \\ \gamma \end{array} \begin{array}{c} \begin{array}{ccc} \text{Distance to CH } (\alpha) & \text{Residual energy } (\beta) & \text{Sensing range overlapping } (\gamma) \end{array} \\ \begin{bmatrix} \alpha/\alpha & \alpha/\beta & \alpha/\gamma \\ \beta/\alpha & \beta/\beta & \beta/\gamma \\ \gamma/\alpha & \gamma/\beta & \gamma/\gamma \end{bmatrix} \end{array}$$

$$= \begin{bmatrix} 1 & 2/1 & 3/1 \\ 1/2 & 1 & 2/1 \\ 1/3 & 1/2 & 1 \end{bmatrix}$$

where the three criteria (distance to CH, residual energy and sensing range overlapping) are denoted by α, β and γ respectively. The selection of these initial values is motivated by our choice that "Distance to CH" is the most important, than "Residual energy" is the next important followed by "Sensing range overlapping" as the least important factor. This choice reflects a typical set of parameters for energy conservation.

The computed eigenvector W = [0.5396 0.2970 0.1634]. It indicates the local weight of the distance to CH, residual energy, and sensing coverage overlapping respectively so that we can see clearly that the distance to CH is the most important criterion, and sensing coverage overlapping is the least. According to Eq. (4), we can get the eigenvalue $\lambda_{max} = 3.0093$. Consequently, consistency ratio can be calculated as $CR = 0.0047 < 0.1$, thus matrix A satisfies the consistency check.

Each sensor node determines the weight matrices of alternatives under three factors and then gets global weight based on its specific situation. Afterwards, its eligibility as a sleeping node can be finally decided.

If there are eight candidate nodes in each cluster, all the eight weight matrixes of alternatives under three factors construct a 8×3 matrix, denoted as $W_{n_i/j}$, i=1, 2, … 8, j=1, 2, 3. The final weight of each alternative is calculated using Eq. (6) with n =3. The larger the final weight of node, the higher the probability of node which is eligible to be put into sleep. Thus, the r fraction of nodes with the largest weight are selected as the sleeping nodes in the current *round*.

5.6.4. Integrated Sleep-Scheduling and Routing Algorithm

In multihop environment, we investigate the sleep scheduling problem as well as multipath routing problem and propose an integrated AHP based sleep scheduling and routing (A-SR) scheme in a ring based multi-hop network topology with the destination (or BS) at the center, as shown in Fig. 3. In the existing multi hop sleep scheduling scheme (e.g., [8]), although routing is integrated, residual energy is the only factor considered. In the existing geographical routing approach (e.g., [19]), the path selection

doesn't consider the remaining battery capacity of each node, which is a very important factor for energy constraint sensor networks. In our A-SR sleep scheduling part, *distance to destination, residual energy,* and *sensing coverage overlapping* are included, with the latter two factors the same as the single hop case. In contrast with the single hop case, the node with shorter distance to destination is preferred to be put into sleep since they are more energy hungry as a relay. In routing part, *distance to the destination, residual energy*, and *queue size* of each sensor node are included, with the former two factors the same as the proposed multi hop sleep scheduling. Our scheme is a fully distributed approach where each sensor only needs the above parameters, and we use AHP to handle these parameters in the A-SR.

In our A-SR scheme, we only keep the second assumption from single hop case. And we need to additionally assume that the event detection by the nodes in the network occurs periodically and all nodes are synchronized. Thus our A-SR can be executed round by round based on the period. For A-SR routing part, the detailed explanations of the three criteria for next hop relay node selection are given as follows:

1) Distance to destination: Distance of a node to BS which is the destination. The geographical location of destination is known to the source node (as in [19]), and the physical location of each sensor node can be estimated easily if the locations of three sensor nodes (within a communication range) are known in a USN. The node with shorter distance to the destination is preferred to be selected.

2) Residual energy: Remaining battery of the sensor node. The energy consumption for transmission and reception can be calculated using Eqs. (7) and (8).

3) Queue size: It indicates the buffer capacity at the node. This parameter helps avoid packet drops due to congestion at the receiver.

The optimized node selection in multipath routing is also a multiple factors optimization problem and can be achieved using AHP.

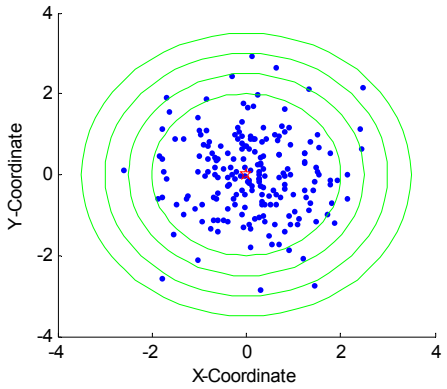
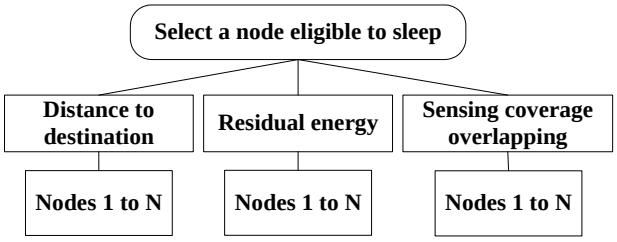
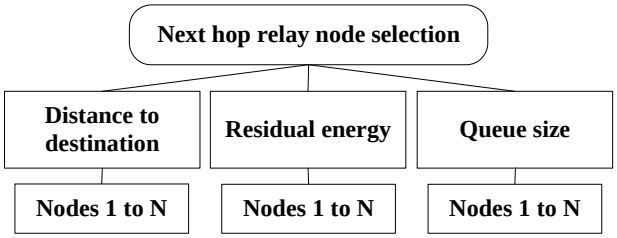


Fig. 3 Ring based multihop network topology.

In our A-SR for M-path routing, the source node select M nodes in its communication range for the first hop relay. Assume there are N ($N > M$) nodes in its communication range, nodes that are farther to the destination node than the source node are not considered. Choosing M nodes from remaining eligible nodes is based on AHP (as will be described in detail). Starting the second hop, each node in the M-path selects its next hop node also using AHP.



(a) AHP hierarchy for sleep nodes selection



(b) AHP hierarchy for next hop relay nodes selection

Fig. 4 AHP hierarchy for decision making in a multi-hop network.

In the AHP hierarchy model, the goal of the decision "next hop relay node selection" is at the top level of the hierarchy as shown in Fig. 4 (b). The next level consists of the three decision factors and at the bottom level there exist the N alternative sensor nodes to be evaluated.

We assume that each sensor node keeps a table which has some information about its neighbor nodes: locations, battery level, and queue size. The table is updated periodically by the locally-broadcasted information (beacon) from each neighbor node. We define a time interval T, during which the three parameters (locations, battery level, and queue size) do not change very much. This time interval T is the shortest time duration that a sensor node will send another beacon. Each sensor examines itself the status of the three parameters in every period T, and if a certain parameter has changed above a threshold, it will locally broadcast a beacon.

In the route discovery phase, the source node uses AHP model to evaluate all eligible nodes (closer to destination) in its communication range based on the parameters of each node: distance to destination, residual energy, and queue size. The source node chooses the top M nodes based on the local weight that this node will be selected. And the source node sends a Route Acknowledgement (RA) packet to each desired node, and each desired node will reply using a REPLY packet if it is available. The structure of RA and REPLY is summarized in Table 3. If after a certain period of time, the source node did not receive REPLY from some desired node, it will pick the node with highest weight

among the remaining *N-M* nodes. In the second hop, the selected node in each path will choose its next hop node using the same process. As illustrated in Fig. 5, node B needs to choose one node from four eligible nodes C, D, E, and F based on their three parameters, and sends RA packet to the selected node and waits for REPLY. If the top one node is unavailable (for exmaple, selected by another path), then the top second node will be selected. Consequently, *M* paths can be set up.

For A-SR sleep scheduling part, we only present its AHP hierarchy model which is shown in Fig. 4 (a), due to the similarity to the single hop case.

For energy analysis, we adopt the previously presented transmission energy model, i.e., Eq. (7). In multihop networks, the energy for reception and data aggregation also need to be taken into account. Thus, to receive an *l*-bit data, the radio expends:

$$E_R(l) = lE_{elec}, \quad (13)$$

and the energy for data aggregation is set as E_{DA} = 5 *nJ/bit*, the same as [21].

In AHP modeling, the evaluation matrices for A-SR, here denoted as A2, is determined based on Eq. (1) as follows:

$$A2 = \begin{array}{c} \\ \alpha \\ \beta \\ \gamma \end{array} \begin{array}{c} \begin{array}{ccc} \alpha & \beta & \gamma \end{array} \\ \begin{bmatrix} 1 & 2/1 & 3/1 \\ 1/2 & 1 & 2/1 \\ 1/3 & 1/2 & 1 \end{bmatrix} \end{array}$$

where the three criteria, shown in Fig. 4 (a) and (b) from left to right, are denoted by α, β and γ respectively.

The computed eigenvector W has the same value as the single hop case since we assumed the same evaluation matrix. So we can observe that the distance to destination is the most important criterion, and sensing coverage overlapping and queue size are the least. We can again get the eigenvalue λ_{max} = 3.0093, and consequently matrix A2 satisfies the consistency check.

Each sensor node determines the weight matrixes of alternatives under three factors and then gets global weight based on its specific situation. Its eligibility as next hop relay node and sleep node can be finally decided by the AHP hierarchy model.

Table 3: RA and REPLY message structure

Type	Desired Node ID	Self Node ID	Dest_X	Dest_Y	Src_ID

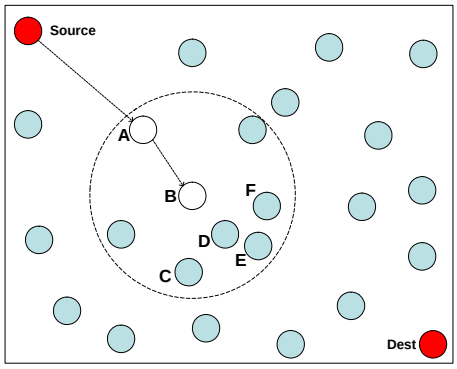


Fig. 5 Illustration of next hop node selection.

5.6.5. Performance Evaluations

In order to evaluate the single hop sleep nodes scheduling scheme by AHP, we compare it with the upper (lower) bound (which optimizes merely the current factor), LDS [7] and random scheduling scheme. In our simulation, the 50*m* by 50*m* square monitored area is assumed. All nodes except CHs equip with identical sensors and the sensing and communication range are equal to 8*m* and 16*m* respectively. Initial energy in each node is 2*J*. We set the total number of nodes to be N_t=50 and number of static clusters to be 2. Thus the number of nodes in each cluster is 25 by assuming a uniform distribution of nodes.

Assume the CH plans to allow 25*r* nodes in its cluster to sleep in each cycle. In the random scheduling scheme, the CH randomly selects *r* fraction sensor nodes to sleep. At first, we compare the average energy consumption in a cluster by AHP based scheme and random scheduling scheme to show the energy that can be conserved by our scheme. Figure 6 provides the energy consumption verses the fraction of sleeping nodes of the three schemes. Furthermore, we also consider the ideal case where Eq. (7) is used to determine the minimum energy consumption which provides a lower bound on average energy consumption. It shows that the energy consumption in case of the proposed AHP based scheme is less than that of the random scheme, however slightly more than that of the LDS. The energy savings can be enhanced with an increasing value of *r*. For an *r* value of "0.7", the energy consumed by the AHP based scheme is 49.3% less than by random scheme and 9% more than LDS.

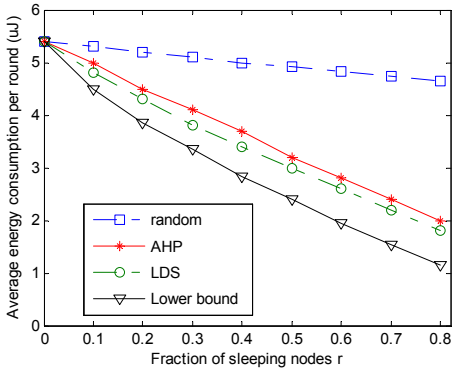


Fig. 6 Energy Consumption in the cluster per round.

Network lifetime can be defined as the time when a fraction of nodes, θ, run out of energy. In Fig. 7, we evaluate the lifetime of the three schemes and the upper bound for various values of *r* andθ. The length of each round is 5 seconds. We can see that the lifetime of both schemes is prolonged with the increasing of *r* and the proposed AHP based scheme greatly outperforms the random scheme and is close to LDS. This is in line with the analysis that the proposed scheme can balance the energy consumption among all the member nodes. We also can see that the lifetime of all the schemes increases with an increase ofθ. This is because the network can be alive up to the time when θ fraction of nodes are drained of their energy.

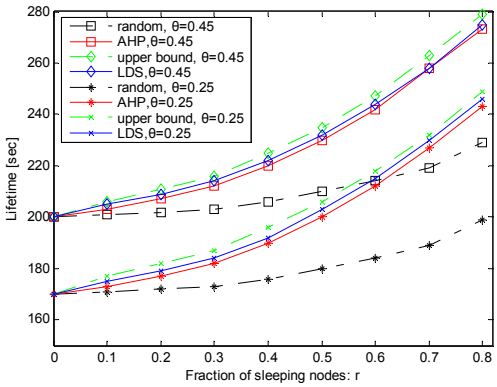


Fig. 7 Lifetime comparision.

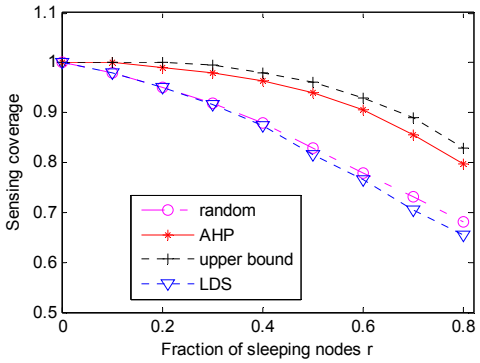


Fig. 8 Coverage verses the fraction of sleeping nodes.

Figure 8 provides the comparison of coverage ratio verses the fraction of sleeping nodes *r*. The coverage here is defined as the ratio of the union of all sensor nodes' sensing areas to the whole monitored environment. For the detailed explanation of coverage ratio calculation method, please refer to [22]. Fig. 8 shows that for the three schemes (AHP, LDS [7] and random) the coverage ratio decreases with the increasing of the fraction of sleeping nodes, *r*. LDS shows similar sensing coverage ratio with random scheme (though different pattern) since the sensing coverage of the LDS scheme in the border

area is lower than that in the central area, as sensor nodes close to the border are put into sleep with higher probability. However, in case of the proposed AHP based sleeping scheme, the coverage ratio still can maintain above the desired value of 0.98 when up to 30% nodes are put into sleep. It indicates that the tradeoff in terms of coverage is not very critical using the AHP based scheme. AHP based scheme outperforms the LDS and is close to the upper bound in that the AHP takes overlapping coverage as one of the impact factors while the LDS does not but only energy saving.

To evaluate the integrated sleep scheduling and routing in multi hop networks by AHP, we have used J-Sim [23] as the simulation environment. 60 sensors are randomly deployed in an area of 100m x 100m. The source and destination sensors are set as 2J initially, and 5 couples of source and destination nodes are communicating at the same time in this network. All the other sensors have initial energy of 0-2J. The buffer capacity of each sensor node has been taken as 5 packets with packet length 512 bit and bit rate 9.6kb/sec. The time interval T is set as 10s in our simulation. The source node select $M=3$ nodes in its communication range for the first hop relay. From the second hop, each node along the three paths selects only one node toward its next hop.

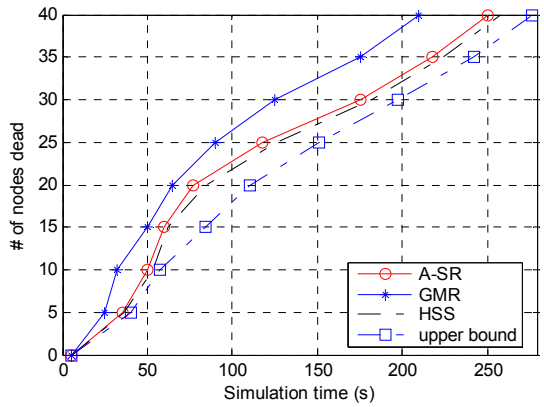


Fig. 9 Lifetime comparison.

We compare our A-SR with Hop-based Sleeping Scheduling (HSS) algorithm [8], upper bound and the geographical multipath routing (GMR) [19] scheme where only distance to the destination is considered. In Fig. 9, we plot the simulation time versus the number of nodes dead. It shows that when 50% nodes (30 nodes) die out, the network lifetime for A-SR has been extended more than 40%. A-SR significantly outperforms GMR and has similar performance to HSS. In Fig. 10, we compare the packet loss rate of these three schemes. Packets are dropped either due to insufficient buffer capacity at the receiver or because of the lack of energy needed to transmit the packet. Observe that our A-SR outperforms the GMR and HSS with about 20% and 10% less packet loss respectively resulting in greater reliability. The average latency during transmission (end-to-end) is 424.23ms for our A-SR, 407.5ms for GMR and 422.8ms for HSS, and link failure rate for A-SR is 6.51%, but for GMR is 10.42% and for HSS is 10%. Due to the integrated sleep scheduling, in our proposed scheme, the network coverage ratio does not drop below the satisfactory value 0.97 when up to around 30% nodes are put into sleep.

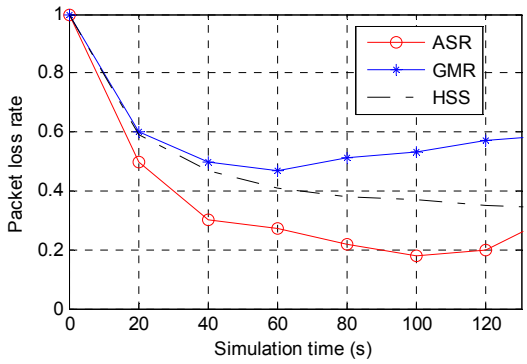


Fig. 10 Simulation time vs. packet loss rate.

5.6.6 Conclusions and Future Work

In this chapter, we proposed a sleeping scheduling scheme in a single hop network and an integrated sleep scheduling and routing protocol in a multi hop network based on AHP. In the single hop network, three factors contributing to the optimal nodes scheduling decision are considered and they are 1) distance to CH, 2) residual energy, and 3) sensing coverage overlapping, respectively. In the multi hop network, our proposed A-SR scheme includes distance to destination, residual energy, and sensing coverage overlapping for sleep scheduling, and distance to the destination, residual energy, and queue size of each sensor node for routing. To evaluate the performance, in the single hop network case, we evaluated the efficiency of our proposed scheme in terms of energy consumption, lifetime and coverage ratio, and compared with the upper (lower) bound, LDS and traditional random sleep scheduling scheme in heterogeneous clustered sensor networks. The proposed scheme was observed to improve network lifetime and conserve energy. We also evaluated the efficiency of the proposed scheme in the multi-hop environment and the results showed that it could extend the network lifetime much longer than the original geographical routing scheme which only considered distance to the destination location, and it had similar lifetime performance with HSS. Moreover, the proposed scheme could reduce the packet loss rate and link failure rate since the buffer capacity was considered. In both single hop and multi hop network environment, the sensing coverage capabilities were not compromised.

In the future work, we may consider the node mobility as another factor for decision making and design such protocol. Moreover, different node distributions will be further considered.

5.6.7 References

[1] Xiaoling Wu, Hoon Heo, Riaz A. Shaikh, J. Cho, O. Chae, and S. Lee: Individual Contour Extraction for Robust Wide Area Target Tracking in Visual Sensor Networks. Proc. of 9th IEEE International Symposium on Object and component-oriented Real-time distributed Computing (ISORC), Gyeongju, Korea (2006) 179-185

[2] Y. Xu, J. Heidemann and D. Estrin, Adaptive energy-conserving routing for multihop ad hoc networks, Research report 527, USC/Information Sciences Institute (2000).

[3] T. L. Saaty, Fundamentals of Decision Making and Priority Theory with the Analytic Hierarchy Process, RWS Publications, U.S.A., 2000.
[4] D. Tian and N. D. Georganas: A coverage□preserving node scheduling scheme for large wireless sensor networks. Proc of the 1st ACM international workshop on Wireless sensor networks and applications (2002) 32 – 41.
[5] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris: Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. ACM Wireless Networks, Vol. 8, No. 5 (2002).
[6] W. Ye, J. Heidemann, and D. Estrin: An Energy□Efficient MAC Protocol for Wireless Sensor Networks. Proc of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), New York, NY (2002).
[7] J. Deng, Y. S. Han, W. B. Heinzelman, and P. K. Varshney: Scheduling Sleeping Nodes in High Density Cluster□based Sensor Networks. ACM/Kluwer Mobile Networks and Applications (MONET) Special Issue on Energy Constraints and Lifetime Performance in Wireless Sensor Networks (2005).
[8] Yun Wang, Demin Wang, Weihuang Fu, and Dharma P. Agrawal: Hops-based Sleep Scheduling Algorithm for Enhancing Lifetime of Wireless Sensor Networks. Proc of IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS), Canada (2006) 709-714
[9] Rick W. Ha, Pin-Han Ho and X. Sherman Shen: SS-trees: a cross-layer organizational approach for mesh-based wide-area wireless sensor networks. Proc. of the 2nd International Conference on Broadband Networks, Vol. 2, Boston, MA, USA (2005) 823-832
[10] C. P. Bhagwat: Highly dynamic destination-sequenced distance vector routing. Proc. of ACM SIGCOMM (1994) 234-244
[11] Ching-Chuan Chiang, Hsiao-Kuang Wu, Winston Liu, Mario Gerla: Routing in clustered multihop mobile wireless networks with fading channel. Proc. IEEE Singapore Intl Conference on Networks (1997)
[12] C. E. Perkins and E. Royer: Ad hoc on demand distance vector routing. Proc. 2nd IEEE Workshop o Mobile Computing Systems and Applications (1999)
[13] D. Johnson and D. Maltz: Mobile Computing. Kluwer Academic Publishers (1996)
[14] S.J. Lee, and M. Gerla: Split Multipath Routing with Maximally Disjoint Paths in Ad Hoc Networks. ICC (2001)
[15] L. Wang, Y. T. Shu, M. Dong, L.F. Zhang, and W.W. Yang: Multipath Source Routing in wireless Ad Hoc Networks. Canadian Conference on Electrical and Computer Engineering, vol. 1 (2000) 479-483
[16] A. Nasipuri, and S.R. Das: On-Demand Multipath Routing for Mobile Ad Hoc Networks. IEEE ICCCN (1999) 64-70
[17] G. G. Finn: Routing and addressing problems in large metropolitanscale internetworks. USC ISI Report ISI/RR-87-180 (1987)
[18] G. Xing, C. Lu, R. Pless, and Q. Huang: On Greedy Geographic Routing Algorithms in Sensing-Covered Networks. ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Tokyo, Japan (2004)
[19] R. Jain, A. Puri, and R. Sengupta: Geographical routing using partial information for wireless sensor networks. IEEE Personal Communications (2001) 48-57

[20] Yaoyao Yin, Juwei Shi, Yinong Li and Ping Zhang: Cluster Head Selection using Analytical Hierarchy Process for Wireless Sensor Networks. IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), Helsinki, Finland (2006) 1-5
[21] Wendi B. Heinzelman, Anantha P. Chandrakasan, and Hari Balakrishnan: An Application-Specific Protocol Architecture for Wireless Microsensor Networks. IEEE Transactions on Wireless Communications, Vol. 1, No. 4 (2002) 660 – 670
[22] Xiaoling Wu, Shu Lei, Yang Jie, Xu Hui, Jinsung Cho and Sungyoung Lee: Swarm Based Sensor Deployment Optimization in Ad hoc Sensor Networks. Proc. of ICESS' 05/ LNCS, Xi'an, China, (2005) 533-541
[23] J-Sim. http://www.j-sim.org/.

5.7 Primary Study of FSO

Free space optics (FSO) and radio frequency (RF) have been widely used in wireless communication. In this work, we compare the technologies to provide system designers useful metrics.

5.7.1 Introduction

Traditional wireless sensor networks are bound by the provable limits in per-node throughput for radio frequency (RF) based communications. Nowadays, there have been increased interests in the development of sensor nodes that can communicate via free space optics (FSO) [1-4]. FSO refers to the transmission of modulated visible or infrared (IR) beams through the atmosphere to obtain broadband communications. In this work, we provide a comparison work of FSO and RF communication. This comparison can provide useful information for system designers.

5.7.2 Comparison of FSO and RF Communication

At lower data rates, RF is excellent at providing coverage due to the scattering and the diffraction of the radio waves, and the sensitivity of the receivers that can be constructed. Channels are robust to being blocked by obstacles. However, higher data rates require higher frequencies. At these frequencies, the radio signal propagation becomes line of sight, and problems become similar to that of using light. Components operating at these frequencies are expensive, and the advantages of radio (coverage, and receiver sensitivity) become less clear.

As an alternative to the radio frequency (RF) technology, free-space optical (FSO) communication emerged as a promising candidate providing broadband, flexible, secure and low-cost communication links between stationary platforms. Applications using FSO have proved its merits [5-7],[9-12]. The comparison is provided in Table I.

Perhaps the greatest advantage of FSO technology is its high throughput. The need for such a channel is evident when considering bandwidth hungry applications. Terabit per second throughputs have been demonstrated under laboratory conditions. In contrast, widespread RF technologies (e.g. 802.11x) are limited to link throughputs on the order of

10s of Mbps across distances of 10s of meters. Besides, the deployment of an FSO link avoids the stringent restrictions on the usage of the limited bandwidth in RF networks. It also avoids interference with existing RF communications infrastructure and is cheaply deployed since no government licensing of scarce spectrum is required.

Table I A Comparison of FSO and RF Communication

Parameters	Radio-Frequency	Optical Communication
Spectrum	2 to 6 GHz	0.8 to 1.5 THz range (IR band)
Capacity	11, 54, 100Mbps,	Up to 10 Gbps, 160Gbps (lab)
Bandwidth	10 – 12 Mbps	200THz (700-1500nm)
Range	20m - 4km	20m – 1.2km
Output Power	5.15-5.25 MHz 50mW; 5.25-5.35 MHz 250mW; 5.725-5.825 MHz 1W.	650nm 5-500mW; 880nm 2.5-500mW; 1310nm 45-500mW; 1550nm 50-500mW
Power Consumption	2.31E-02 (J/Mb)	2.00E-03 (J/Mb)
Power Loss [8]	2.4 GHz 100dB/km; 915 MHz 92dB/km; 5.7 GHz 108dB/km.	Clear 5-15dB/Km; Rain 20-50dB/Km; Snow 50-150dB/Km; Fog 50-300 dB/Km
Security	Low	High

The main limitation of FSO is the requirement that a direct line-of-sight path exists between a sender and a receiver (e.g. flying objects such as birds may block the transmission). Another main problem of FSO is that the availability of services is weather sensitive (e.g. heavy rain, fog, snow or strong wind).

There are also some other constraints: eye safety limitation (IEC60825-1,Amendment 2), SNR varies significantly with the distance and ambient noise, accurate alignment of transmitter-receiver necessary, etc. However, researchers have been working to enhance its performance in various ways, e.g. with multibeam architectures, larger power margins, backup systems, etc.

5.7.3 Conclusion

Free Space Optics is being increasingly considered as an attractive option for the rapid provisioning of multi-gigabit per second links. In our work, we provide a comparison between FSO and RF technologies. The main contribution is to provide useful metrics for system designers. The metrics can be adopted as a method for quantitatively defining system performance.

5.7.4 References

[1] J. M. Kahn, R. H. Katz, and K. S. J. Pister, “Next century challenges: Mobile networking for “smart dust””, in Proc. ACM/IEEE International Conference on Mobile Computing and Networking, Seattle, Washington, August 15-19, 1999, pp. 271–278.

[2] B. A. Warneke, M. D. Scott, B. S. Leibowitz, L. Zhou, C. L. Bellew, J. A. Chediak, J. M. Kahn, B. E. Boser, and K. S. J. Pister, “An autonomous 16mm3 solar-powered node for distributed wireless sensor networks,” in Proc. IEEE Sensors, Orlando, Florida, January 2002, pp. 1510–1515.
[3] W. Mao and J. M. Kahn, “Free-space heterochronous imaging reception of multiple optical signals,” IEEE Transactions on Communications, vol. 52, no. 2, pp. 269–279, February 2004.
[4] J. Llorca, A. Desai, U. Vishkin, C. Davis, and S. Milner, “Reconfigurable optical wireless sensor networks,” in Proc. SPIE vol. 5237, Optics in Atmospheric Propagation and Adaptive Systems VI, J. D. Gonglewski and K. Stein, Eds., Barcelona, Spain, February 2004, pp. 136–146.
[5] “A Fiber-optic Powered Wireless Sensor Module Made on Elastomeric substrate for Wearable Sensors”, the 26th Annual International Conference of the IEEE EMBS,2004
[6] D-Link Corporation, “DLINK 802.11(g) PCMIA card,” www.dlink.com, 2005
[7] FreeScale Corporation, “XS110 UWB datasheet,” www.freescale.com, 2005
[8] Cambridge Silicon Radio, “CSR Bluecore2 datasheet,” www.csr.com, 2005
[9] “Optical wireless communication in distributed sensor networks”, The International Society for Optical Engineering
[10] Dominic O’Brien, “optical wireless communications challenges and prospects”, University of Oxford.
[11] “Urban Optical Wireless Communication Networks” IEEE Communication Magazine, February 2003
[12] Brian J. d'Auriol and Tanushree Ghosh, "A Systems Model for Computation, Communication, Command and Control (C4) in a Spacecraft or Satellite Cluster", Proceedings of The International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT) 2006, December 4-7, 2006, Taipei, Taiwan, pp. 285-290, IEEE Computer Societ

6. Autonomic Context Aware Middleware

6.1 Background and Motivation

6.1.1 Middleware

In a distributed computing system, middleware is defined as the software layer that lies between the operating system and the applications on each site of the system.

Stimulated by the growth of network-based applications, middleware technologies are taking an increasing importance. They cover a wide range of software systems, including distributed objects and components, message-oriented communication, and mobile application support.

Companies and organizations are now building enterprise-wide information systems by integrating previously independent applications, together with new developments. This integration process has to deal with legacy applications. A legacy application can only be used through its specific interface, and cannot be modified. In many cases, the cost of rewriting a legacy application would be prohibitive.

An increasing number of systems are composed of a collection of various devices interconnected by a network, where each individual device performs a function that involves both local interaction with the real world and remote interaction with other devices of the system. Examples include computer networks, telecommunication systems, uninterruptible power supply units, decentralized manufacturing units.

Users interact with Internet applications through a variety of devices, whose characteristics and performance figures span an increasingly wide range. Between a high performance PC, a smart phone, and a PDA, the variations in bandwidth, local processing power, screen capacity, ability to display color pictures, are extremely large [1].

6.1.2 Functions of Middleware

In all of the above situations, applications use intermediate software that resides on top of the operating systems and communication protocols to perform the following functions:

- Hiding distribution, i.e. the fact that an application is usually made up of many interconnected parts running in distributed locations;

- Hiding the heterogeneity of the various hardware components, operating systems and communication protocols;

- Providing uniform, standard, high-level interfaces to the application developers and integrators, so that applications can be easily composed, reused, ported, and made to interoperate;

- Supplying a set of common services to perform various general purpose functions, in order to avoid duplicating efforts and to facilitate collaboration between applications.

These intermediate software layers have come to be known under the generic name of middleware [1].

6.1.3 Setting and History behind this project

Our main goal is to provide user-friendly information and communication services, anywhere and anytime.
For this purpose we were focusing with central Middleware for ubiquitous application. Figure 6.1 shows our previous approach. We were using sensors to collect data and send the raw data to central middleware, which is a PC equipped with middleware software, middleware than process data and send to application.

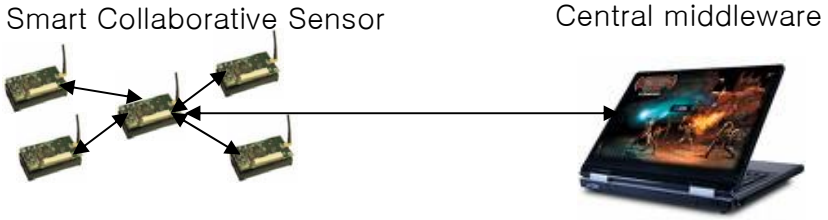


Fig. 6.1: Previous approach– Central middleware, only sensors was used.

A major challenge in developing sensor network systems and algorithms is that transmitting data from each sensor node to a central processing location may place a significant drain on communication and energy resources. Such concerns could place undesirable limits on the amount of data collected by sensor networks. However, in many applications, the ultimate objective is not merely the collection of "raw" data, but rather an estimate of certain environmental parameters or functions of interest (e.g., source locations, spatial distributions). One means of achieving this objective is to transmit all data to a central point for processing. An alternate approach based on distributed in-network processing which, in many cases, may significantly decrease the communication and energy resources consumed [2].

The basic idea is illustrated by a simple example. Consider a network comprised of n sensor nodes and one central server uniformly distributed over a square meter, each of which collects m measurements. Suppose that our objective is to compute the average value of all the measurements. There are three approaches one might consider:

1. Sensors transmit all the data to a central server which then computes the average. In this approach O(mn) bits need to be transmitted over an average of O(1) meter.

2. Sensors first compute a local average and then transmit the local averages to a central server which computes the global average. This requires only O(n) bits to be transmitted over O(1) meter.

3. Construct a path through the network which passes through all nodes and visits each node just once. The sequence of nodes can be constructed so that the path hops from neighbor to neighbor. The global average can be computed by a single accumulation process from start node to finish, with each node adding its own local average to the total along the way. This requires O(n) bits to be transmitted over only O(n-1/2) meters.

Clearly the last procedure could be much more communication efficient than the other two approaches. It is also clear that a similar procedure could be employed to compute any average quantity (e.g., a least squares fit to any number of parameters)[2].

In our previous approach we were focusing with second procedure. But in order to achieve third procedure central middleware is not enough.
For this purpose we proposed a new distributed middleware architecture such that,

- Application supported by distributed middleware can do complex, fast operation.
- Facilitates integration of components in distributed heterogeneous environment.
- Optimized / Faster Integration.
- Faster system development.
- Cost-effective operation and management.
    - Autonomic Capabilities (self-defining, self-configuring, self-optimizing, self-healing, context-aware and anticipatory) are closely related to those systems.
    - Middleware services and architectures are gradually evolving to support autonomic computing systems

Fig 6.2 shows our current distributor middleware approach. Middleware is integrated into Smart Sensor, Smart Object, Handheld and PC.

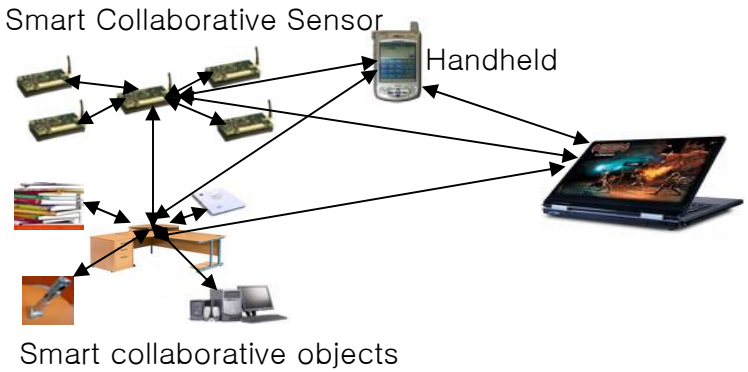


Fig. 6.2: New approach– each of the cooperative items has a middleware

6.1.4. Goal of the project

Our goal is to development of distributed middleware which facilitates Ubiquitous Application.

- Allowing the service developer to focus on the service logic rather than the middleware implementation,
- It will deal with the challenges like resource constrains, mobility, heterogeneity.
- Integration of Sensor, Smart Object, handheld devices into smart environment.
- Integrate Collaborative Middleware to all the components associated with the environment.
- Using handheld to access the mobile infrastructure.

6.1.5. Scope of the project

Scope of the project is pointed below,

- Middleware covers all the devices and network connected to the Ubiquitous sensor network.
- Embed smartness to cooperating items.
- Dynamic power management.
- Reduce the communication between cooperating items.
- Event based communication.
- Application knowledge in nodes will be imposed during design phase.
- Automatic configuration of nodes and error handling.
- Support for time and location management.

6.2 Middleware Architecture

We propose distributed middleware architecture based on aCAMUS, which is autonomic by nature and supports applications where sensing and acting devices themselves drive the network behavior. In our proposed architecture application is not confined in a powerful node, rather it is deployed on the devices embedded within the physical environment. We discuss our middleware design principles in subsequent sections.

6.2.1. Simplicity and Flexibility

Our framework comprises of several types of middlewares with different capabilities to be deployed in different objects such as sensors, smart objects, handheld devices and central server. Therefore, we name these middleware as follows:

- Sensor Node Middleware (SNM)
- Smart Object Middleware (SOM)
- Handheld Device Middleware (HDM)
- Backend Server (aCAMUS)

However simplicity as well as flexibility in the design of these middlewares is inevitable. Hence, we keep the overall architecture, modules and their interfaces same in all middlewares. Middlewares differ only in capabilities of their respective modules. In our current design we consider three different computing resources (e.g. CPU, Memory, Power) of the entities (e.g. sensors, smart objects, handheld devices, backend server etc.) based on which we limit the capabilities of these middlewares. Figure 6.4 shows our common middleware architecture named aCAMUS.

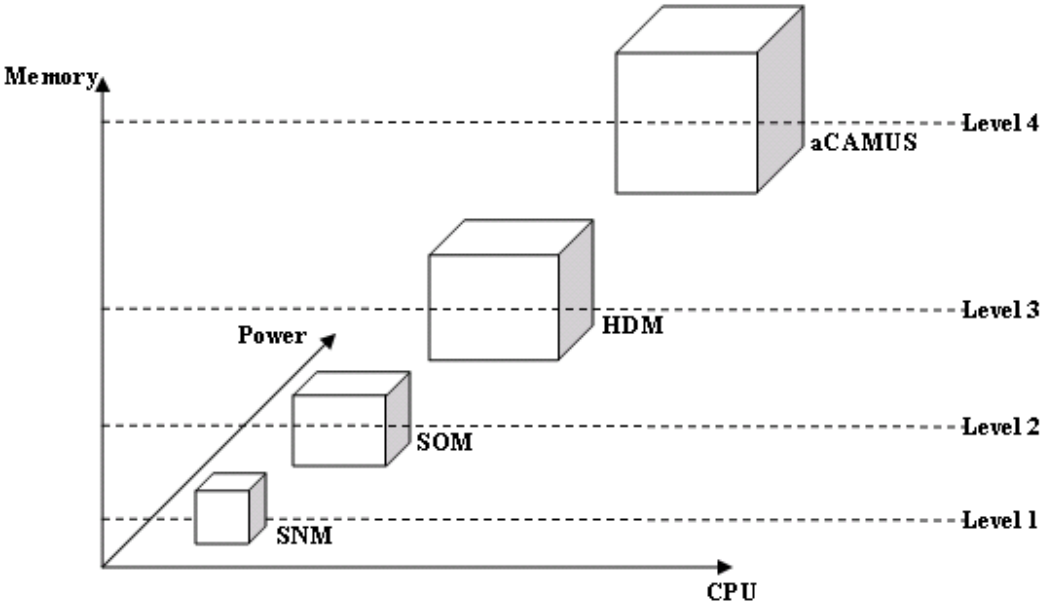


Fig 6.3 Different middlewares and their relative levels. All middlewares have the similar architecture but have different capabilities w.r.t computing resources

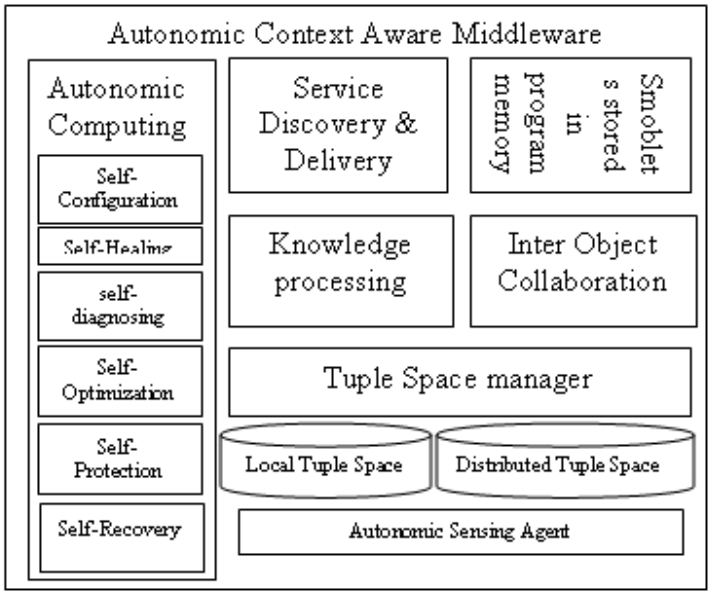


Fig 6.4 Common Middleware Architecture (i.e. aCAMUS)

6.2.2. Pluggable and Upgradeable Modules

All module of a middleware are pluggable and hence easily upgradeable. Autonomic manager module helps to upgrade the modules on the fly. Hot-swapping mechanism is also desired. OSGi framework may be a good choice for this purpose. Thus even after deployment we may upgrade some reasoning module when necessary. Besides, if computing resources for a certain entity (e.g. sensor, smart object etc.) changes, we change or limit the module's capabilities accordingly. For example, if CPU, memory is increased in sensors we may choose more efficient but resource consuming reasoning algorithm in SNM and vice versa.

6.2.3. Vertical Context Propagation

Middlewares generate contexts after collecting context data from the lower level middlewares. For example, SNM generates context from sensory data, but SOM generate context based on contextual data from SNMs. Each middleware type can generate context with certain accuracy. This is because lower level middleware does not have efficient reasoning mechanism as of upper level middlewares. Hence, we assume that context flows only in vertical direction from bottom to top. For example, context from level 1 middleware (e.g. SNM) propagates to Level 2 middleware (e.g. SOM) and so on. However, if immediate upper level middleware is not available, next higher level middleware becomes a candidate for receiving context from this lower level middlewares. On the other hand, client applications can discover and collect context from middlewares of any level depending on clients' need.

6.2.4. Horizontal Collaboration

As middleware have different reasoning capabilities, we assume that middlewares collaborate with other middleware of the same level to generate contexts of certain

accuracy. However, middlewares collects data from lower level middlewares/ sensors and then sends generated context to upper level.

6.3. Sensing Agent

The purpose of sensing agent is to retrieve the sensory data with the help of appropriate OS function calls and send these raw data to Tuple Space manager. Sensor access strategies determine when to read out sensors. With respect to the time period between consecutive samplings and the time at which to access sensors, we distinguish between the following strategies:

Best effort: Some sensors, for example accelerometers and microphones, monitor environmental parameters that can change rapidly in relatively short time frames. In such cases, the best effort strategy allows an application to access sensors as often as technically feasible in order to monitor environmental changes accurately. The above-mentioned function for reading out a sensor is thereby called **whenever possible**.

Fixed effort: In this sensor access strategy, there is a fixed time period between consecutive sensor samplings. For example, the statement every 2000 ms means that the corresponding sensor access function is scheduled to be called every two seconds. There are two ways to implement the fixed strategy on a microcontroller. The first, and more precise one, is to use a timer and to access the sensor from the corresponding interrupt handler. This ensures that an application accurately conforms to the given time interval. In our prototypical implementation, however, we have implemented a scheduler that repeatedly checks for functions that need to be executed. The accuracy of such an approach is influenced by the execution time of other scheduled functions.

Random interval: In order to avoid unwanted synchronization effects between nodes, it is sometimes necessary to vary the time period between consecutive sensor accesses. Such a behavior is supported by the random interval strategy, where a time interval is specified out of which a value is randomly chosen. The statement random [1000, 2000] ms is an example of this where the time between consecutive readings is randomly distributed between one and two seconds.

Event-driven: In contrast to the need for regular sensor access, some sensors only need to be read out under special circumstances – i.e., in case of certain internal or external events. Event-driven sensor access can considerably reduce the number of times a sensor needs to be sampled. It can therefore contribute towards reduced energy consumption, as the process of sampling sensors often consumes significant amounts of energy. Hence, especially when designed for resource restricted smart objects; applications should aim at accessing sensors as seldom as possible. Event-driven sensor access is specified using adaptation rules, and not directly in sensor access statements.

6.4. Inter Object Collaboration

The **Inter Object Collaboration** handles all basic communication-related issues required for smart object collaboration. The main purpose of this module is to

1. perform the collaboration between smart objects in order to achieve consensus free or consensus based operation,

2. integrate handheld (e.g. PDA) into environment of cooperating smart everyday objects and

3. delegate a task to neighboring entity (smart object or handheld). Estimates computational power needs by other smart objects

We will explain each of the scenarios in details in the following sections.

6.4.1. Collaboration between smart objects/sensors

Depending on the application requirements, collaboration may follow a consensus-free and/or a consensus-based approach.

Consensus-free — entities may need to take local decisions hence they do not negotiate a common decision and communicate different information, i.e., sensory data, fused information, context/sub context and next action(s) in order to aid this local decision making.

In figure 6.6 we are showing an example of consensus-free operation between smart objects. In this figure we have 4 smart objects SO1, SO2, SO3, and SO4. SO1 is the master node or full function device (FFD) and SO2 to SO4 are the slave nodes or reduced function devices (RFDs). Each of the Smart Objects contains a distributed tuple space as shown in tables the first column of the table is the tuple name, second one is the actual value and the third one is the epoch number. The responsibilities of FFD are to perform as task and/or delegate a task to neighboring entity. In consensus-free operation FFD performs a simple task (F). That is, F = (A+C-D) * D + E.

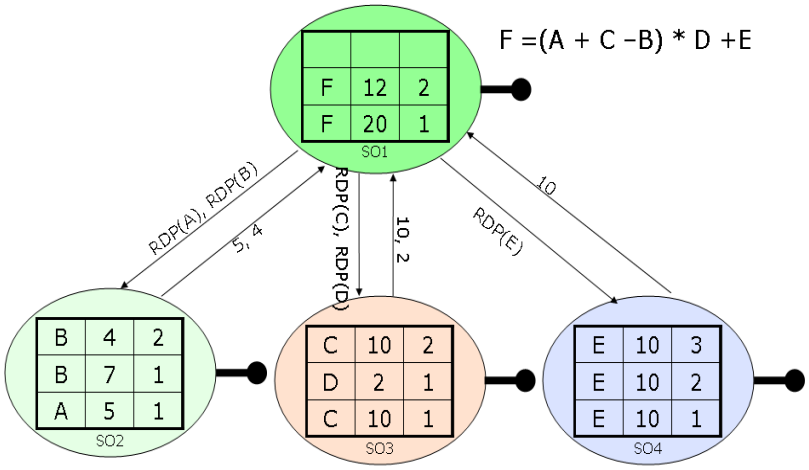


Figure 6.6: Example of consensus-free operation between smart objects.

In order to perform this task FFD communicates with other RFDs for A, B, C, D and E. Here, SO2 contains of A and B, SO3 contains C and D and SO4 contains E. FFD executes a read command and upon receiving read command RFDs send their latest values to FFD. And FFD computes F as shown in the figure 6.7.

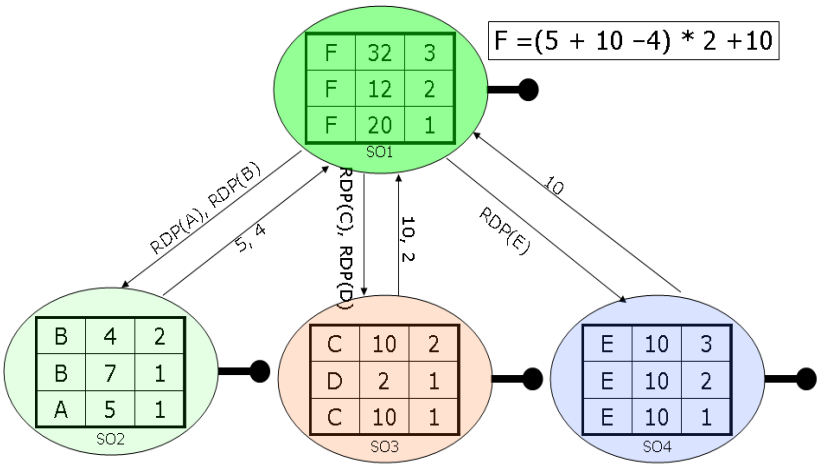


Figure 6.7: Example of consensus-free operation between smart objects.

Consensus-based — a system/sub-system may need negotiated outcomes; consequently entities are compelled to communicate different information in order to take a common decision.

In figure 6.8 we are showing same example as shown at consensus-free operation but the operation is consensus based. SO1 delegates SO2 to computing (A + C – B). SO1 first get C, D from SO3 and E from SO4 and sends the values to SO2. SO2 than compute the

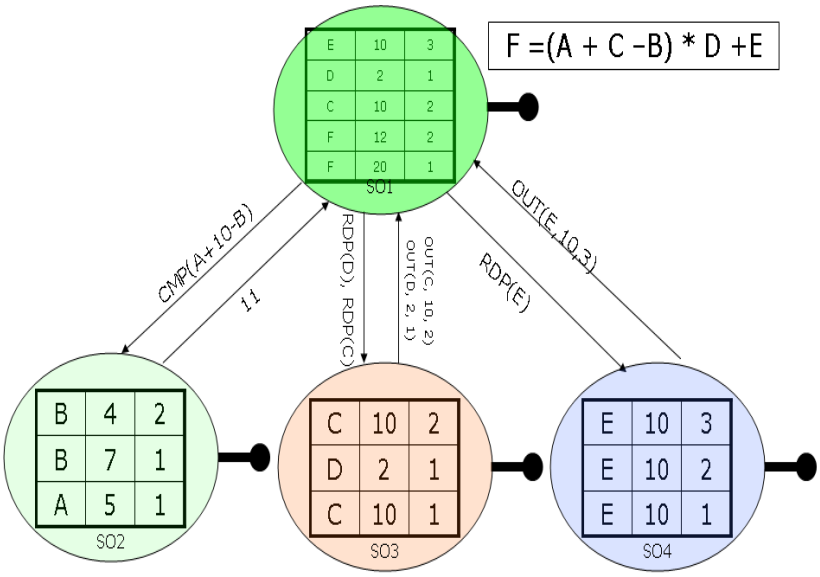


Figure 6.8: Example of consensus-based operation between smart objects.

desired function and send the value to SO1. And SO1 computes F as shown in figure 6.9.

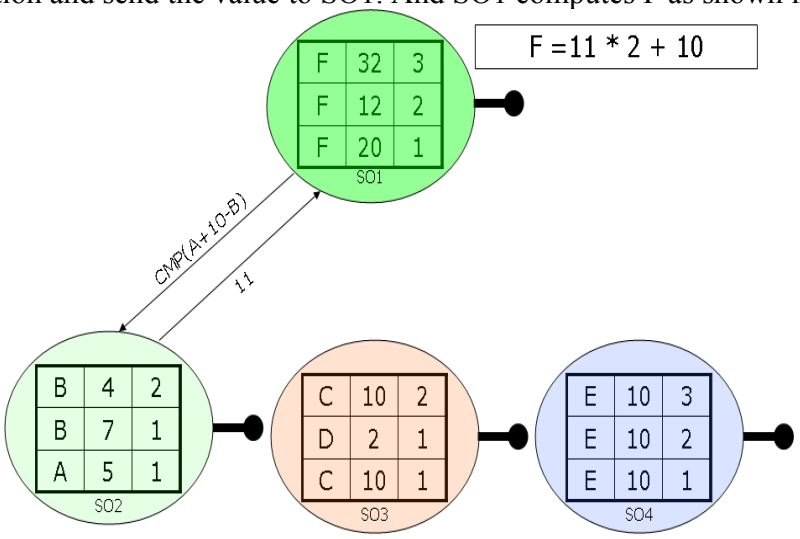


Figure 6.9: Example of consensus-based operation between smart objects.

6.4.2. Integrating handheld into environment [2]

In this section we describe how smart everyday appliances can be used in collaboration with handheld in order to perform context aware computing.

The purposes of integrating handheld into our environment such that,

1. It provides user interfaces for SO.
2. Mobile infrastructure access point.
3. Handle large amounts of sensory data to provide context-aware services.

Integrating handhelds into environments of cooperating smart objects in a software framework called *Smoblets[2]*. The term *Smoblet* is composed of the words *smart object* and *Applet*, reflecting that in our approach active Java code is downloaded from smart objects in a similar way in which an Applet is downloaded from a remote Web server [2]. Figure 6.10 shows the incorporation of smoblets into the CAMUS proposed architecture and is adapted from [2].

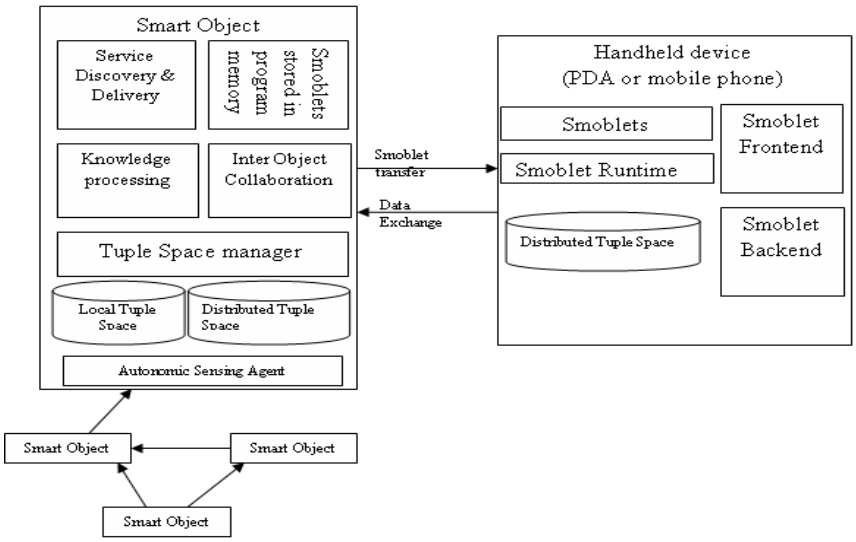


Figure 6.10: Overview of Smoblet System[2]

About smoblets:

- The code that is transferred to a nearby user device - referred to as Smoblet
- Consists of Java classes that are developed on an ordinary PC during the design of a smart object.
- As the smart objects themselves cannot execute Java code, these are transferred to the handheld to execute and transfer back the result.

The Smoblet backend is responsible for executing downloaded code on a handheld device. It also protects the user device from malicious programs and enables downloaded Java classes to access data on other platforms by providing an interface to the distributed tuplespace implementation. In contrast, the Smoblet front-end helps users to search for smart objects in vicinity, to explicitly initiate download, and to customize the behavior of the Smoblet backend system.

An example of user interfaces for Smart Object (SO):
Figure 6.11 shows an example of user interface for SO provided by PDA. The purpose of this example is to show how we plan to use handheld as a complex user interface. In this example we show how we can monitor patient's heart beat in a smart hospital/home. The data will be gathered by a smart object capable of sensing heart beat. The gathered data will than be sent to nearby handheld along with smoblet that will be responsible for displaying the sensed data.

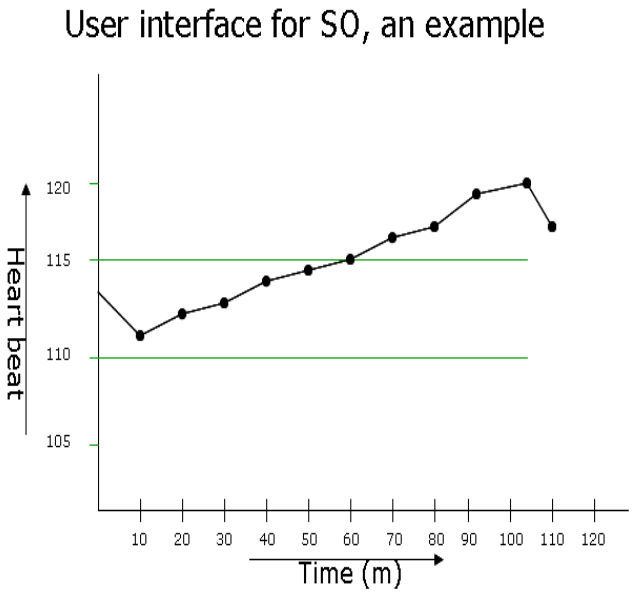


Figure 6.11: An example of user interface for SO provided by PDA

An example of mobile infrastructure access point:
Figure 6.12 shows an example of how we can use handheld as a mobile infrastructure access point. In this scenario we consider smart collaborative objects (SCO) environment to monitor patients in a smart hospital.

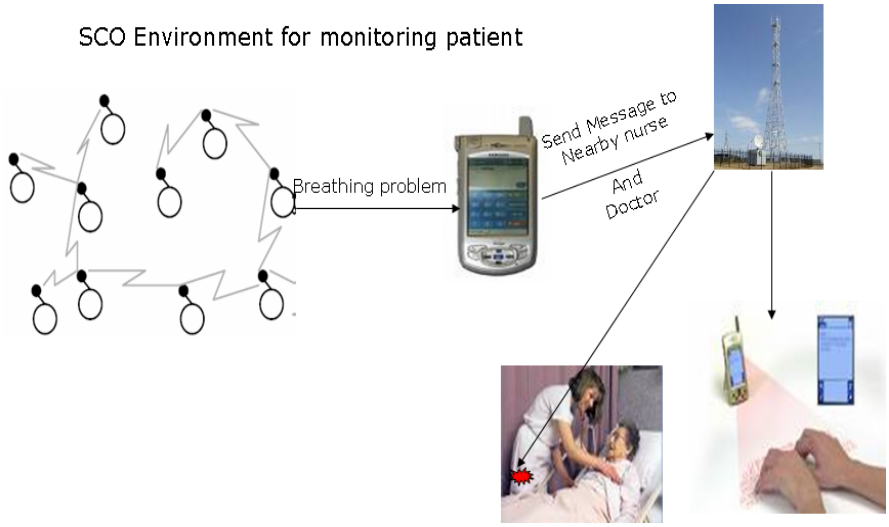


Figure 6.12: An example of mobile infrastructure access point

A patient John equipped with of smart objects and is staying in a room of a smart hospital. John experiences difficult breathing, his smart objects, given his medical history, triggers an alert that this is a serious condition; notifications need to be sent to the assigned nurse who is watching another patient and to assigned doctor who is working at her office in the next block. In order to send such notifications smart hospital uses existing mobile

infrastructure. That is, using the John's mobile, SCO send alert message to both doctor and nurse.

6.5 Knowledge processing

In essence a distributed architecture for collaborative objects presents a challenging scenario for implementing knowledge-processing mechanisms that can provide the same level of functionality as in the case of a centralized architecture. The report is organized as follows initially we will tackle the issues related to a distributed knowledge-base in section II and in section III we will deal with reasoning mechanisms that can be efficiently implemented in collaborative smart objects.

Details about Knowledge Processing will be found at chapter 8.

6.6 Adaptive, Scalable and Proactive Context delivery

The purpose of this module is to,

1. Provides scaleable context discovery and delivery mechanism.
2. Classifies the HOT and COLD context data.
3. Send it to multicast or unicast channel.
4. Proactive (prediction of future request).

In a ubiquitous environment, numerous clients will request for different contexts. The context provider should be scalable in the sense that it should be able to satisfy the need of context information among large number of clients without performance degradation. On the other hand due to the existence of mobile devices, number of clients as well as their request for context may change dramatically. Hence the system should be adaptive to the number of requests. Scalability may be achieved through broadcasting the items. But that may consume valuable bandwidth with less important data. On the contrary, pure unicast can not achieve scalability as in this case their server communicates with each of the client independently. Thus to achieve the scalability and adaptability in context delivery we have incorporated hybrid dissemination approach, where hot item (most requested) are broadcasted and the cold items are unicasted. But this approach introduces the challenge of identifying hot and cold items. Tough request count is a good candidate to calculate the hot item, but that may lead to starvation of delivery of cold item (with fewer requests). To cope up with this problem we need to take account of the waiting time along with the number of requests. This approach is also known as RxW algorithm. Thus we give higher priority to the item with highest number of request or the item with longest waiting time. We further try to perform optimal bandwidth division between broadcast and unicast to reduce average latency in receiving the context [3].

Figure 6.13 shows an example of delivering a context in multicast channel. In this scenario we consider smart collaborative objects (SCO) environment in a smart seminar room of a smart hospital. Dr. Shannon wants to consult about the current situation of

patient John whose health condition decrease over night with senior doctors, Dr. Frank and Dr. Stella and assigned nurse. She calls for a seminar while she is in her office. While she reaches to the seminar room, associated smart objects determine that it is about to start the seminar and sends a context to all the mobile devices in a multicast channel such that the devices do not make sound.

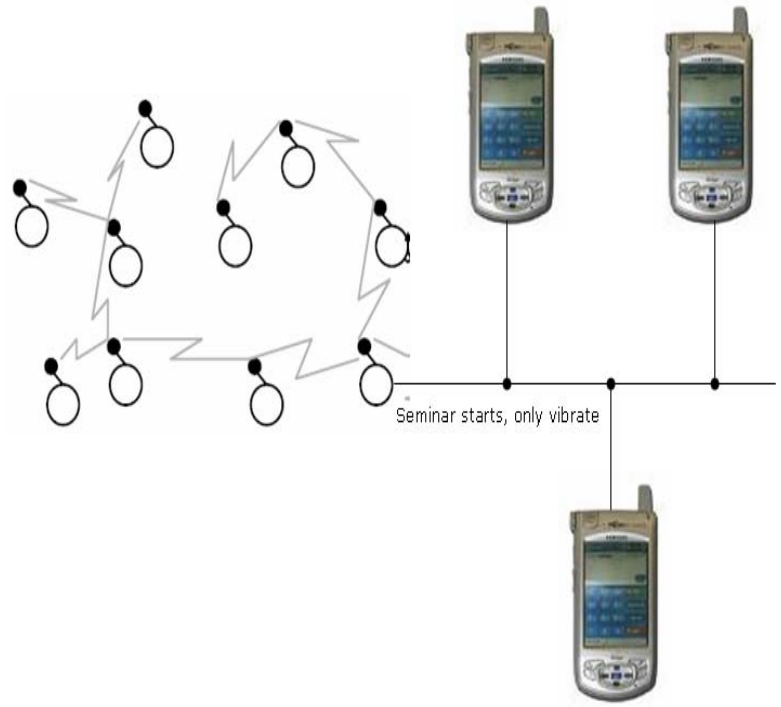


Figure 6.13: An example of delivering a context in multicast channel

6.7 Project plan

In this section we described our project plan for the next 5 years along with summary of methodology we are going to use.

6.7.1 Summary of methodology

- What general development approach will be used?
  - We will be using Object Oriented Approach for,
    - Faster development.
    - Reuse of previous development.
    - Increased quality.
    - Modular architecture.
    - Better mapping to problem domain.

- How will the project team be organized?
  - Size of the development team will be 6.
  - The change control board will consists of 3 people.
- What development and collaboration tools will be used?
  - Project website

  - Project mailing lists
  - Issue tracking system
  - Version control (VC) system
- How will changes be controlled?
  - Requests for requirements changes will be tracked in the issue tracker
  - The change control board (CCB) will review requested changes and authorize work on them as appropriate
  - After the feature complete milestone, no new features will be added to this release.
  - After the code complete milestone, no entirely new product source code will be added to this release.
  - All source code commit log messages must refer to a specific issue ID, after the feature complete milestone.
- How will this plan be updated?
  - Throughout the project.
  - Will be placed on VC.
  - Automatic notification to be sent to a project mailing list.

6.7.2 Work breakdown structure and estimates

Step	Description	Estimate (Month)
1.	**Background Study and Report**	**9**
1.1	Scenario analysis	3
	Fundamental scenario analysis	2
	Supporting scenario analysis	1
1.2	Survey	5
	Existing middleware architecture for SCO, SCS, handheld and backend server.	3
	Existing tools.	2
1.3	Report writing	1
2.	**Design**	**9**
2.1	Middleware architecture design	4
	Define module functionalities	3
	Interface specification	1

2.2	Inter middleware communication	4
2.3	Documentation	1
3	**Implementation**	**18**
3.1	sensor node middleware(SNM).	2
3.2	smart object middleware(SOM).	3
3.3	Integration of SNM and SOM	1
3.4	handheld device middleware (HDM).	5
3.5	Integration of SNM, SOM and HDM	1
3.6	backend server (aCAMUS).	4
3.7	Integration of SNM, SOM, HDM and aCAMUS	2
3.8	Update of Documentation	
4	**Testing**	**12**
4.1	Test bed design and Implementation	3
	Port SNM to sensors.	0.5
	Deployment of sensors.	1
	Port SOM to SO.	1
	Deployment of SO	0.5
4.2	Scenario testing	8
	Collaboration among sensor	1
	Collaboration among sensor and smart object	1
	Collaboration among sensor and smart object and Handheld	1
	Collaboration among all the entities.	4
4.3	Update of Documentation	1

5.	**Refinement and finalization**	**8**
5.1	Revise design	4
5.2	Revise Implementation	3
5.3	Retesting	1
6	Documentation and final report	4

6.8. References

[1] What is middleware? http://middleware.objectweb.org/

[2] Michael Rabbat and Robert Nowak: "Distributed Optimization in Sensor Networks" http://www.ece.wisc.edu/~nowak/dosn.pdf

[2] Frank Siegemund and Tobias Krauer, "Integrating Handhelds into Environments of Cooperating Smart Everyday Objects"

[3] Lenin Mehedy: Scalable, Semantic and Proactive Context delivery for Ubiquitous Computing. Thesis for the Degree of Master of Science. Department of Computer Engineering, Kyung Hee University, Seoul, Korea.

7. Ubiquitous Database Manager

7.1 Introduction

There exists numerous low-level context and high-level context in Ubiquitous Environment. Context Repository is a persistent storage of the contexts and support query to applications. The context repository can be classified into two types: Centralized Repository and SCO Repository. Centralized Repository is placed at backend server which has enough computational power and large capability storage. On the other hand, in Smart Collaborative Object (SCO) environment, there is a repository at handheld devices which has a small computational power and storage space. Therefore the repository must be light weight and the contexts should be stored in a distributed space because of the characteristics of handheld devices, sensors and objects in SCO environment.

7.1.1 SCO Environment

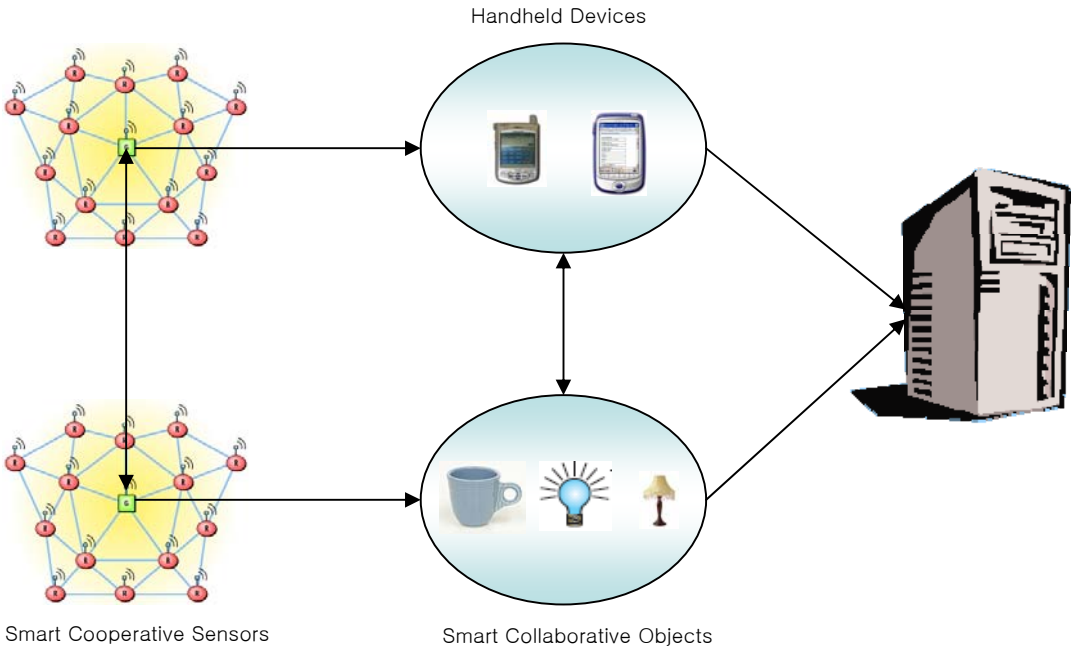


Figure 7.1: Smart Collaborative Object Environment

Figure 7.1 shows a typical example of smart collaborative object (SCO) environment. There are some passive sensors connected to active sensors. These active sensors can co-operate with each other. The term 'co-operate' means that they can communicate with each other by means of sharing information. These smart co-operative sensors can also communicate with other smart objects and handheld devices. Again, smart object and handheld devices can communicate with each other. The contexts generated from users' activities will be stored in the backend server as well as some important contexts will be stored in smart objects and handheld devices. These contexts contain useful information regarding users or these are frequently shared contexts.

7.1.2 Our Objectives

Our main concern is to build a context repository for the SCO environment. In the course of building this context repository we need to meet the following objectives too. These are listed below.

1. Providing a means for context storing in smart objects and smart sensors using tuple space.

2. Ontology driven, lightweight context storing mechanism in users' handheld devices with agent based communication.

3. Build a central context repository in backend server to support complex reasoning and querying.

7.1.3 Challenges

The design and development of a context repository for smart collaborative object environment faces a number of challenges. These challenges are:

1. *Low power and limited memory of handheld devices*: The main goal of smart collaborative object environment is to provide services to the users in a context-sensitive way. It is very much expected that user will have handheld devices like PDA, mobile phones those have limited power resource and memory. Therefore, the context storage should be light weight so that the most essential information and context should be stored in users' handheld devices.

2. *Location-centric*: The entities in SCO environment are mobile in nature. This mobility adds location as a new dimension to applications that does not typically play a role in stationary scenarios. Consider a system that can answer questions such as "find the drugstores within 2 miles of my current location". Such a system must track the location of the current user and be able to access information based on relative locations and distances.

3. *Machine learning*: Whether the system obtains its context information from sensors, user input, PIM (personal information management) applications, or some combination of these, it must perform a good deal of processing over the data in order to be able to accurately assess the state of the environment and the intensions of the user. Thus, context-aware applications impose demanding requirements for machine learning techniques.

4. *Context reasoning*: This is very much necessary to get high level context from low level context for effective context-sensitive information in SCO environment. The provision of context reasoning will be extremely efficient in order to be able to interact with the user in a useful and unobtrusive manner.

5. *Diversity of contexts*: In SCO environment, there should be a lot of heterogeneous devices. Those devices will produce a number of contexts which will be different in

nature. For designing and developing a successful context repository in SCO environment we need to identify some common framework to store that context information. This common framework should be understandable by all the devices and agents interacting in the system.

6. *Objects are distributed*: Smart collaborative objects will be distributed over a particular region. Users will be also roaming around over that region. Therefore centralized storage system will be in no use in this environment. We must have to design the context repository in distributed manner to satisfy the queries issued by the distributed objects and users.

7.2 Context Modeling and Representation

7.2.1. Context Modeling Approaches

Context Modeling is one of the most important job in designing an application for a smart space and also a really hard job that related directly to the collaboration between smart objects. In a context modeling survey, Strang and Linnhoff-Popien specified some important requirements that we should consider when modeling context for ubiquitous computing system in general.

1. Distribution composition: ubiquitous environment makes the composition and administration of the context model and its data varies with notably high dynamics

2. Partial validation: partial validation on structure and instance level is highly necessary to control modeling intention error-prone due to the complexity of contextual interrelationships

3. Richness and quality of information: quality and richness of information varies over time and is different from various sensors. A context model should support quality and richness indication

4. Incompleteness and ambiguity: a context model should cover the incomplete and/or ambiguous contextual information that gather from sensor network.

5. Level of formality: A precise and traceable representation of contextual facts and interrelationships helps every party in ubiquitous computing interaction share the same interpretation of the data exchange and its meaning.

6. Applicability to exiting environments

Based on these requirements, these two authors summarized the most relevant modeling approaches classified by the scheme of data structures which are used to exchange contextual information. Then they evaluated these approaches against the requirement specified above.

Figure 7.2: Environment variables: key-value pairs

1. *Key-value models*: The simplest data structure that represents the context information by a value provided via an environment variable to an application (Figure 7.2). In distributed service framework, a service is described by a set of these simple attributes. The service discovery procedure then operates and matching algorithm on these attributes. This model is easy to manage but inefficient in describing complex contextual information and weak to response to all requirements above. There is no common scheme or definitions to check against.

2.*Markup Scheme Models*: All markup based models (Figure 7.3) use a hierarchical data structure consisting of markup tags with attributes and content. *Profiles* represent typical markup-scheme models. Typical examples for such profiles are the Composite Capabilities / Preference Profile (CC/PP) [3] and User Agent Profile (UAProf), which are encoded in RDF/S. various other examples can be found in Strang and Linnhoff-Popien (2004). Markup scheme models are strong concerning the partial validation due to the existence of scheme definition and validation tools. Other requirement is met depending on each specific application.

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:cscp  = "http://example.org/CSCPProfileSyntax#"
  xmlns = "http://example.org/SessionProfileSyntax#"
  xmlns:dev = "http://example.org/DeviceProfileSyntax#"
  xmlns:net = "http://example.org/NetworkProfileSyntax#">
  <SessionProfile rdf:ID="Session">
    <cscp:defaults rdf:resource=
      "http://localSessionContext/CSCPProfile/previous#Session"/>
    <device><dev:DeviceProfile>
      <dev:hardware><dev:Hardware>
        <dev:memory>9216</dev:memory>
      </dev:Hardware><idev:hardware>
    </dev:DeviceProfile></device>
  </SessionProfile>
</rdf:RDF>
```

Figure 7.3: CSCP instance based on RDF

3. *Graphical models*: Context is modeled by using a very well know modeling tool Unified Modeling Language due to its generic structure. Various approaches exist where contextual aspects are modeled in by using UML, e.g. Bauer in [5]. Another modeling approach includes an extension to the ORM (Object-Role Modeling) by context information presented in Hendricksen et al. [6]. This kind of approach is particularly useful for structuring, but usually not used on instance level and useful for deriving an

ER-model so that can be structure in a relational database. Other requirements are considered but not fully address.

4.*Object Oriented Models*: It uses the power of object-oriented benefits (e.g., encapsulation, reusability, inheritance) to represent different context types (such as temperature, location, etc.) and encapsulate the details of context processing and representation on an object level. Access the context and the context processing logic is provided by well-defined interfaces. A representative is the *cues* [7] and *Active Object Model* [8]. Object oriented models are strong at distributed composition. Applicability to existing environment put the burden on the resources of the computing devices that often can not be fulfilled in ubiquitous computing systems.

5.*Logic based models*: This model defines context as fact, expression together with rules on which new concluding expressions or facts can be derived from a set of existing facts and expressions. Contextual information adding, updating and deleting from logic based system in terms of facts or inferred from rules in the system. The strongest point of this approach is high degree of information. McCarthy's *Formalizing Context* [9-10] and Akman&Surav's *Extended Situation Theory* [1] are examples of this approach. This approach is high in formality but difficult to maintain partial validation and has not considered the quality requirement as well as incompleteness and ambiguity. Applicability also leave a major issue

6.Ontology based models: Ontologies represent a description of the concepts and relationships. Therefore, ontologies are a very promising instrument for modeling contextual information due to their high and formal expressiveness and the possibilities for applying ontology reasoning techniques. Various context-aware frameworks use ontologies as underlying context models.

The evaluation of each modeling approach based on the six requirements from Strang and Linnhoff-Popien [2] show that ontologies are the most expressive models and can fulfill all of their requirements. That is why this approach currently emerges as the most appropriate modeling tools for contextual systems.

7.2.2 Context Representation

To be shared, the ontologies need a representation language. Languages like XML that define structure of a document, but lacks semantic model, are not enough for describing ontologies- intuitively an XML document may be clear, but computers lack the intuition. In recent years ontology languages based on Web technologies have been introduced. DAML+OIL [12], which is based on RDF Schema [13], is one such language. It provides a basic infrastructure that allows machines to make simple inferences. Recently, DAML+OIL language was adopted by World Wide Web Consortium (W3C), which is developing a Web Ontology Language (OWL) [11] based on DAML+OIL. Like DAML+OIL, OWL is based on RDF Schema [13], but both of these languages provide additional vocabulary—for example relations between classes, cardinality, equality, richer typing of properties, characteristics of properties, and enumerated classes—along with a formal semantic to facilitate greater machine readability. The OWL language has a

quite strong industry support, and therefore it is expected to become a dominant ontology language for the Semantic Web.

7.2.3.1 RDF Schema (RDFS)

RDFS provides the framework to describe application-specific classes, their properties, and relationships. RDF works on instance level where as RDFS works on schema level. An example is given below.

```
<rdfs:Class rdf:ID="animal" />
  <rdfs:Class rdf:ID="horse">
     <rdfs:subClassOf rdf:resource="#animal"/> </rdfs:Class>
```

7.2.3.2 Web Ontology Language (OWL)

OWL is built on the top of RDF & RDFS. It provides much more expressive than RDF & RDFS with hierarchies and relationships between resources.
It has many predefined classes and properties for ontologies that can be reused. It supports a wide variety of development tools.

7.3 Context Storing

In this section we will discuss on context storing mechanism. Before going into the details, let's have a look on the following questions and answers.

1.Why do we embed smartness to objects?
  Adaptation to users by sensing and computing *current context*

2. How does smart object cooperate with each other?
  By sharing *contextual information*

3. How does smart object adapt to user's current activity?
  By analyzing *current context*

4. How can the system predict user's future behavior?
  Complex reasoning over *historical context*

Therefore, it is quite obvious that we have to store context to achieve our goal. We must have to build the context repository for smart sensors, smart objects, handheld devices as well as backend server. In the previous section, we pick ontology oriented approach as the most suitable one for modeling and representing contextual information. We will discuss the advantages of ontology oriented approach for context storage and ontology based database in coming sections.

7.3.1 Ontology-oriented Approach for Context Storing

Ontology is a data model that represents a domain and is used to reason about the objects in that domain and the relations between them. Ontologies are used in artificial intelligence, the semantic web, software engineering and information architecture as a form of knowledge representation about the world or some part of it. It is a data model

used for implementing semantic web that is a vision of web pages that are understandable by computers, so that they can search websites and perform actions in a standardized way. Semantic web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. The motivation of using ontology driven approach for context repository is three folds.

• Knowledge sharing: Ontology provides a common framework for information sharing. We used to define standard vocabularies at the time of ontology modelling. These set of standard vocabularies could be easily understood by different smart collaborative objects present in the environment. Therefore, the sharing of knowledge could be served in a meaningful way.

• Knowledge reuse: we can build large-scale context ontology by reusing knowledge of several domain specific ontologies. We need not to start from scratch to build up the large-scale context ontology. Objects in the environment can also reuse the knowledge as it is providing a common framework to define contexts.

• Logic inference: Systems built on the top of SCO environment need inference mechanism. Generating high level context from low level context is very much important for smart collaborative object environment. This inferring mechanism helps the proper execution of the system that it gives necessary context-sensitive information to the user thus providing necessary services to users.

7.3.2 Ontology-based Database

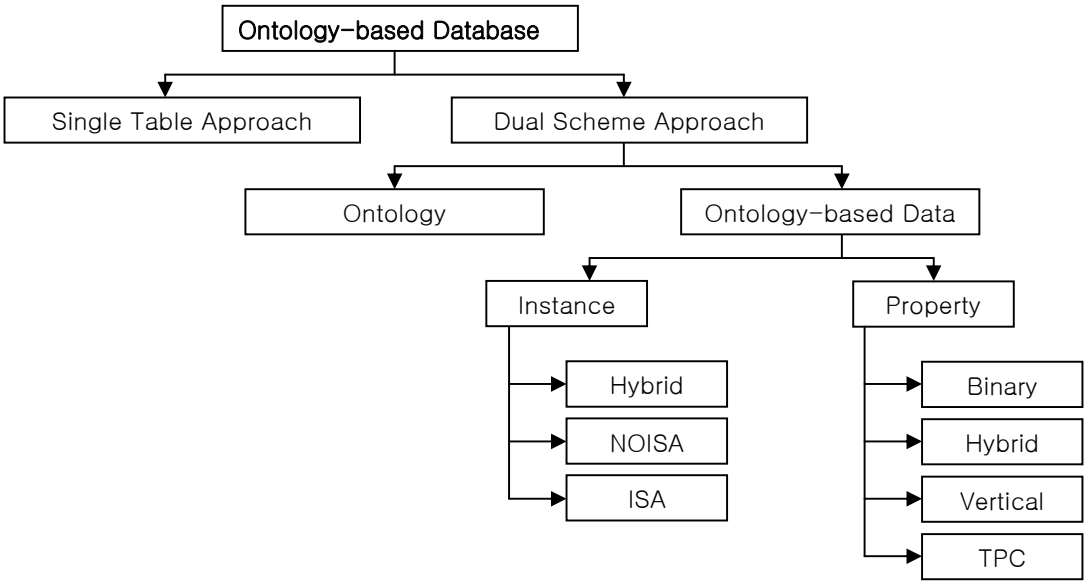


Figure 7.4: Different Schemes for Ontology-based Database

The database that stores data and the ontologies describing their meanings in the same database, known as Ontology-based database (OBDBs). Figure 7.4 shows different schemes used so far. Ontology-based data represents ontology individuals. As for

example, instance of ontology classes. To ensure a high performance of queries on top of OBDBs efficient representation of ontology-based data is needed. There are two main representation schemes have been proposed. They are:

- Single table approach
- Dual schems approach

In the **single table approach** [14-17], the description of classes, properties and their instances are stored in a single table called vertical table [14] (see Figure 7.5). The demerits of single table approach are

- Each column shall be indexed.
- The property column shall be clustered.
- Not efficient to process queries having a large number of join operations.

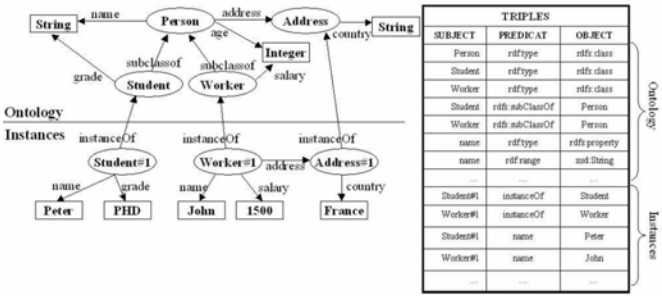


Figure 7.5: Vertical table approach

To overcome the drawback of the first approach, a **dual scheme approach** (see Figure 7.6) has been proposed. It consists in storing separately ontologies and instance data in two different structures, called ontology and data, respectively [18-20].

RANGE

TYPE	PROPERTY
xsd:string	1
xsd:integer	2
4#CLASS	3
xsd:string	4
xsd:float	5
xsd:string	6

PROPERTY

ID	NAME
1	name
2	age
3	address
4	grade
5	salary
6	country

DOMAIN

PROPERTY	CLASS
1	1
2	1
3	1
4	2
5	3
6	4

CLASS

ID	NAME
1	Person
2	Student
3	Worker
4	Address

SUBCLASS

super	sub
1	2
1	3

SUBPROPERTY

super	sub

Figure 7.6: Ontology schema in the dual scheme approach

In the dual scheme approach, instances and their properties values are also stored separately (see Figure 7.7). Three different schemes have been proposed to record class

belonging. In the two first approaches, each class is mapped onto a table. The third one, called, hybrid maps all classes on the same binary table.

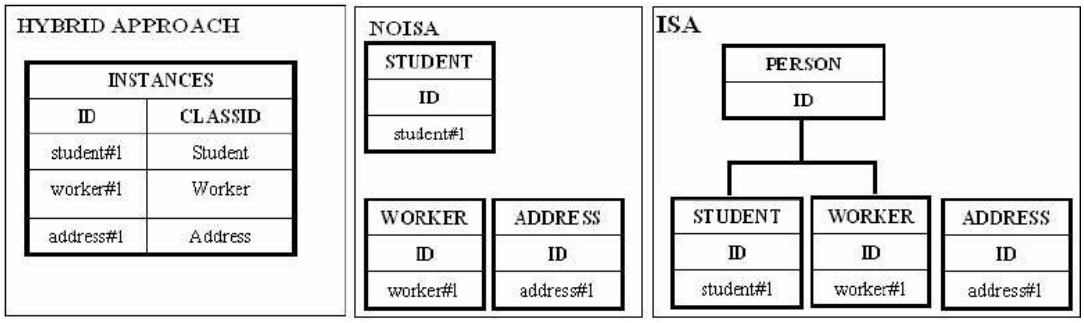


Figure 7.7: Instance scheme alternative representations

Three representations of property values are possible: (1) binary tables, (2) vertical table of triples and (3) hybrid approach (see figure 7.8) consisting of a set of triple tables (one per a range data type).

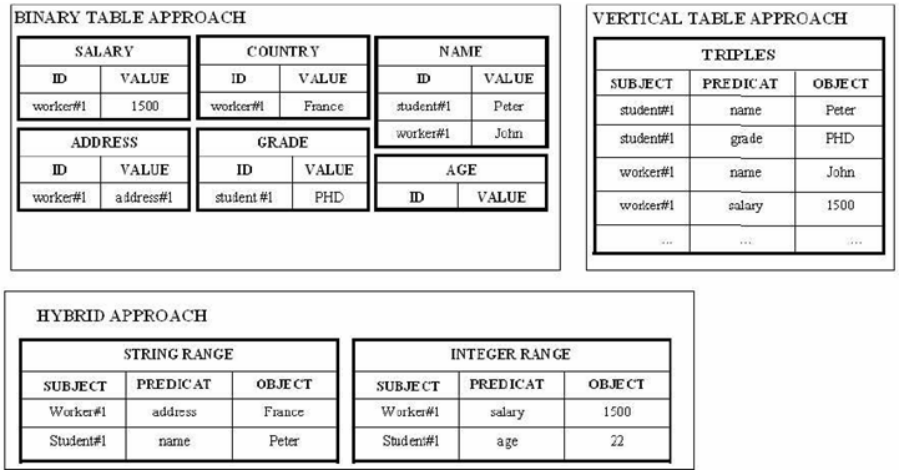


Figure 8: Property value scheme alternative representations

Another new technique of storing data has been proposed named as **table per class representation approach** [21]. This schema consists of all the class applicable properties that are used at least by one instance of the class (see Figure 7.9).

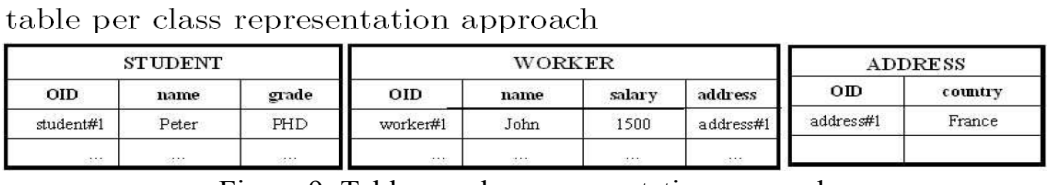


Figure 9: Table per class representation approach

Path based query [22] classifies three types of query patterns

- Queries based on one or multi step structure of triples
- Queries for class hierarchy and property hierarchy

- Queries considering both RDF and RDF schema

An example of the proposed storage structure for path based query has been given below (Figure 7.10).

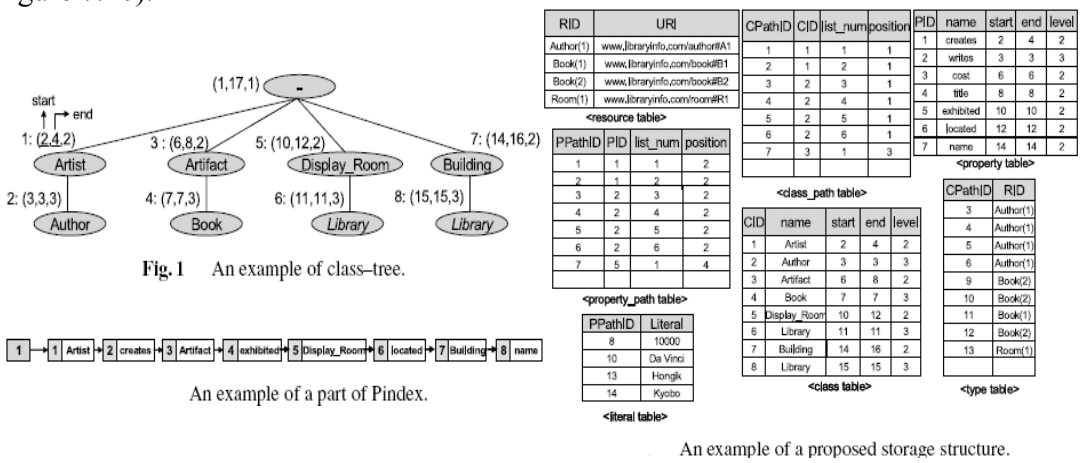


Figure 7.10: Storage structure for path based query

Context Oriented RDF Database [23] is a service satisfying the requirements for CODBS (context oriented database service) based on the semantic web technologies. Here, a profile is referred to a reusable resource (Figure 7.11) that can be identified via a URI. Profiles are containers of key-value pairs. Each pair is a context element where the key serves both as the URI and source of semantics (Figure 7.12).

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns="http://www.zgdv.de/CORD/schemas/UserProfile#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xml:base="cord://st.zgdv.de:999/profiles/me">
  <UserProfile rdf:about="#">
    <personalInfo>
      <PersonalInfo rdf:about="#personalInfo">
        <name>
          <PersonName rdf:about="#personalInfo.name">
            <first>Mohammad-Reza</first>
            <last>Tazari</last>
            <nick>Saied</nick>
          </PersonName>
        </name>
      </PersonalInfo>
    </personalInfo>
  </UserProfile>
</rdf:RDF>
```

Figure 7.11: Partial RDF representation of a user profile

Key	Value
cord://st.zgdv.de:999/profiles/me#	The whole profile resource
cord://st.zgdv.de:999/profiles/me#rdf:type	http://www.zgdv.de/CORD/schemas/UserProfile#UserProfile
cord://st.zgdv.de:999/profiles/me#personalInfo	The embedded resource rooted at 'personalInfo'
cord://st.zgdv.de:999/profiles/me#personalInfo.rdf:type	http://www.zgdv.de/CORD/schemas/UserProfile#PersonalInfo
cord://st.zgdv.de:999/profiles/me#personalInfo.name	The embedded resource rooted at 'name'
cord://st.zgdv.de:999/profiles/me#personalInfo.name.rdf:type	http://www.zgdv.de/CORD/schemas/UserProfile#PersonName
cord://st.zgdv.de:999/profiles/me#personalInfo.name.first	Mohammad-Reza
cord://st.zgdv.de:999/profiles/me#personalInfo.name.last	Tazari
cord://st.zgdv.de:999/profiles/me#personalInfo.name.nick	Saied

Figure 7.12: Set of context elements resulting from figure 8

7.4 Our Solution for Context Repository

For building a successful context repository, we must have to provide a means to interchange contextual information among smart sensors, smart objects. Furthermore, we have to ensure that some user-oriented context will be stored in users' handheld devices as well as each and every context will be stored in the backend server. In the coming portion, we will talk about our solution involving distributed tuple space for smart objects and sensors, ontology-based lightweight database for users' handhelds and context repository for backend server.

7.4.1 Distributed Tuplespace for Smart Objects and Smart Sensors

Distributed tuplespace (Figure 7.13) provides a platform for storing and exchanging sensory data and contextual information. It distributes the originally centralized tuplespace structure among different nodes. In our approach, we will distribute tuplespace among smart objects.

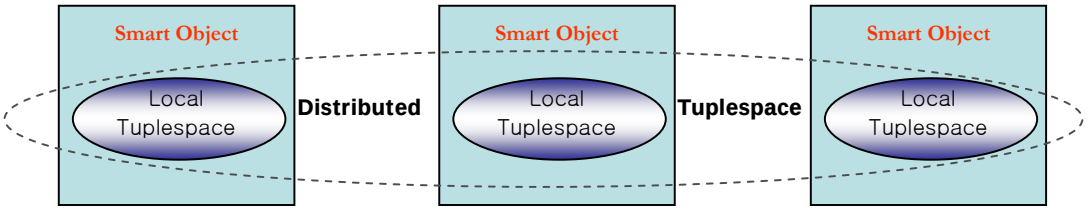


Figure 7.13: Distributed Tuplespace

Here are some more details on distributed tuplespace.

• Tuple spaces are a realization of the associative memory aka the blackboard architecture for storage [24].

• A data structure shared by all the objects cooperating with each other.

• Shared data structure.

• Each node contributes a portion of its local memory to the shared data space.

• Operations on shared data space reflects to all local spaces.

7.4.2 Ontology-based Lightweight Database for users' handhelds

The main component of our ontology driven context repository is **entity**. User, meeting room, class room, restaurant etc. is examples of different kinds of entities. In our definition of an entity computer or any other computational devices are not an entity itself rather it belongs to the environmental context of that particular entity. The contextual information for a particular entity can be roughly subdivided into the following categories.

• Environmental context: This part captures the entity's surroundings like things, services, agents, environmental condition, users and information accessed by users.

• Spatio-temporal context: This type of context is concerned with attributes like time, location and movement.

• Task context: This describes what the entity is doing, it can describe the entity's goals, activities etc.

This contextual information can be efficiently managed by ontology. **Ontology** provides a common framework for understanding this information. Here we give an example of ontology with related concepts for an entity (Figure 7.14).

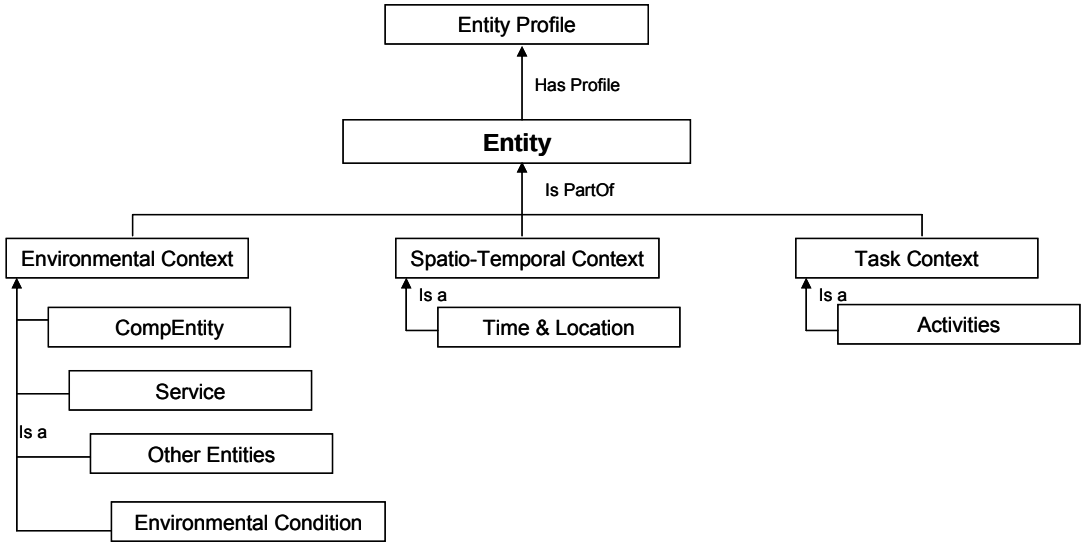


Figure 7.14: Concepts of ontology for an Entity

Each entity in our smart collaborative object environment will communicate with each other by means of its agent. The goal of agents is to reduce user work and information overload.

An **agent** is an autonomous computer program, which has some specific functions and responds to specific events, based on pre-defined knowledge rules or user's instructions [25].

Three fundamental roles of agents (Figure 7.15) are essential to smart collaborative environments: information finder/filter, information interpreter, and decision maker.

• **An information finder and filter** helps users to find out the requested information and filter out unnecessary information according to a specified user task. The agent will provide a reasonable number of choices to users or suggest an alternative option if the requested information items are not available.

• **An information interpreter** can access and convey information from one side to the other. In distributed network environments, heterogeneous data models and systems can not communicate directly.

• **A decision maker** can make decision autonomously based on its own knowledge and user-defined rules. An agent can collect and analyze information according to specific events, such as the migration and linkages of objects and components, and then make an optimal decision based on the rational rules defined by users or other agents.

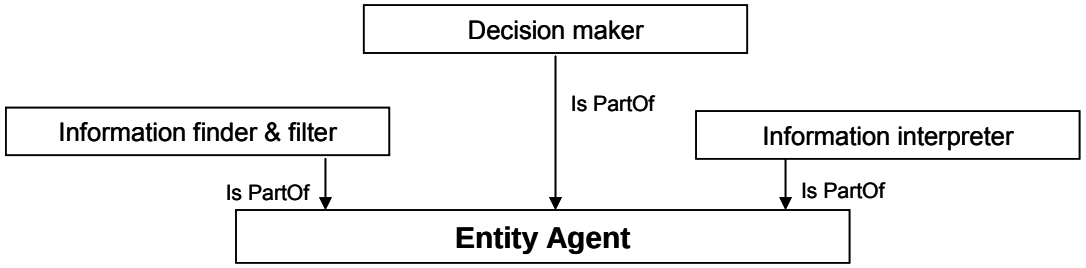


Figure 7.15: Block diagram of an Entity Agent

Agent for a particular entity is responsible for knowledge sharing with other agents of the smart collaborative object environment. Actually the agent retrieves contextual information from the ontology based database of each entity. As our primary concern is to make the database a lighter one, we will use one of the **light-weight database** techniques. This light-weight database holds the ontology and ontology-based data for a particular entity. By means of agent of that entity, this data can be shared with other agents thus leads to the collaboration among different entities (Figure 7.16).

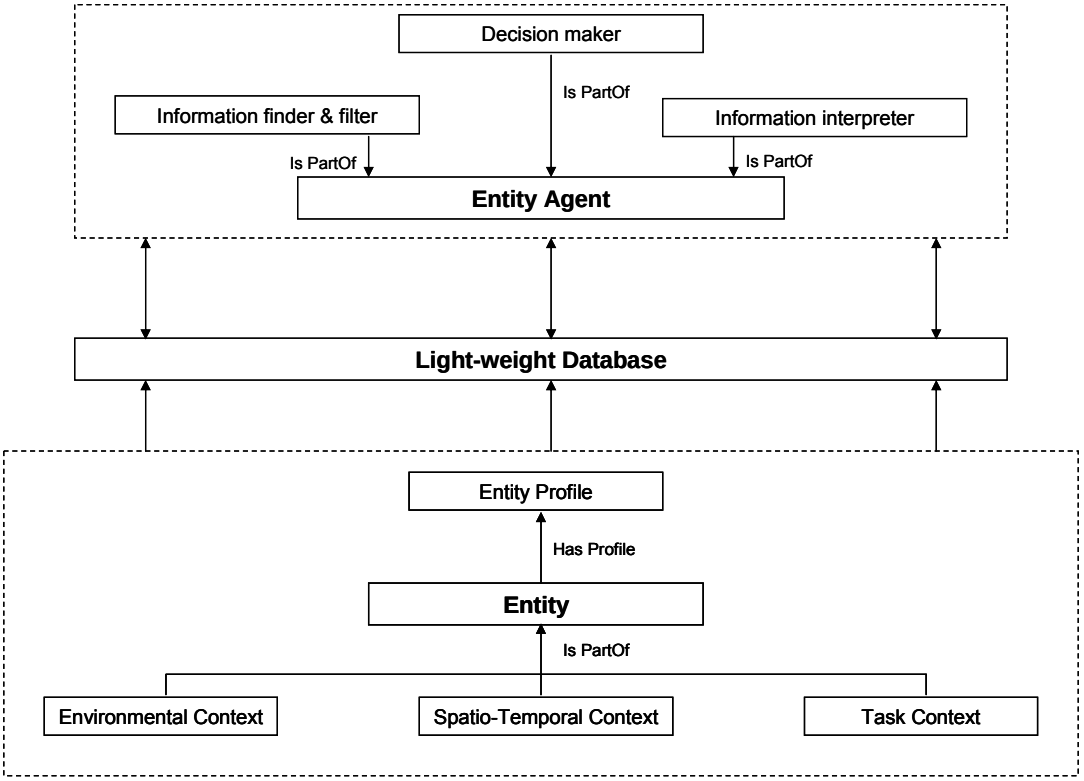


Figure 7.16: Interaction among Entity, Database and Agent

Next, we discuss on some light-weight database techniques like SQLite, HSQLDB, Mini SQL etc.

- **SQLite** [26]: SQLite is a small C library that implements a self-contained, embeddable, zero-configuration SQL database engine. SQLite is an embeddable database system that uses flat files. It does not need to be started, stopped, configured, or managed like other SQL databases. It is lightweight, fast, and compact. And it works completely out of the box without any configuration.

- **HSQLDB** [27]: HSQLDB is the leading SQL relational database engine written in Java. It has a JDBC driver and supports a rich subset of ANSI-92 SQL (BNF tree format) plus SQL 99 and 2003 enhancements. It offers a small (less than 100k in one version for applets), fast database engine which offers both in-memory and disk-based tables and supports embedded and server modes.

- **Mini SQL** [28]: Mini SQL, or mSQL, is a lightweight database engine designed to provide fast access to stored data with low memory requirements. As its name implies, mSQL offers a subset of SQL as its query interface.

7.4.3 Context Repository in Backend Server

There are two alternatives for context storing in backend server.

1. Ontology-based Database

2. Profile-based Storage

Both of these schemes have been discussed on section 3.

In the backend server, context will be collected in a timely manner from smart sensors, smart objects as well as users' handheld devices. Some principle features of backend server includes

- Support complex reasoning (e.g. Bayesian reasoning).

- Support prediction of user and object behavior.

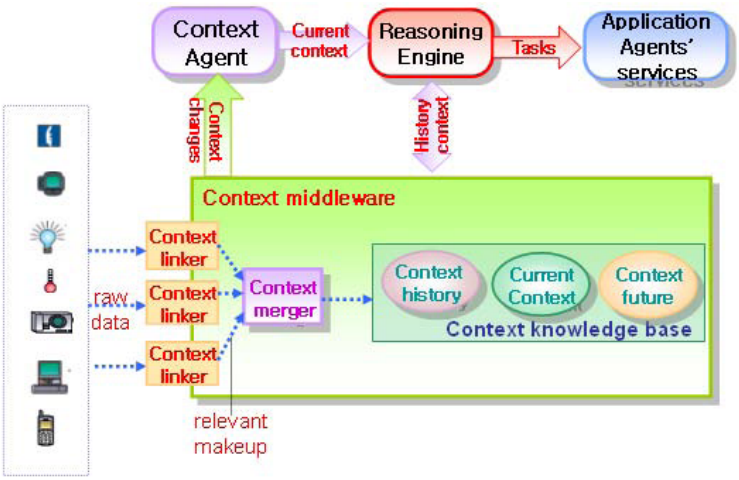


Figure 7.17. Context Repository Architecture in Backend Server

Figure 7.17 depicts the context repository architecture in backend server. Context linker links data from different sources to the relevant information entity (e.g. Context location linker, user linker, device linker). Context merger integrates context from different sources, also identifies which parts of context is changed. Agent using context will subscribe the notification schedule to context middleware and will receive notifications when relevant part of context is changed. While receiving new context, it sends to reasoning engine to specify the tasks needed to automatically perform for users to adapt new context. Tasks will be sent to appropriate application agents' services.

7.5 Context Querying Interface

Our next concern is to how queries will be executed on stored context? Synchronous or asynchronous query mode? What would be the result format and query languages?
For finding the answers for the aforementioned questions, we need to define a query interface. Here we will discuss on SQI (Simple Query Interface) [29].

7.5.1 Requirements and Design Principles

Here we present an Application Program Interface (API) for querying learning objects repositories. Since one major design objective is to keep the specification simple and easy

to implement, the interface is labelled Simple Query Interface (SQI). The collaborative effort of combining highly heterogeneous repositories has led to the following requirements:

1. SQI is neutral in terms of results format and query languages: The repositories connecting via SQI can be of highly heterogeneous nature: therefore, SQI makes no assumptions about the query language or results format.

2. SQI supports Synchronous and Asynchronous Queries in order to allow application of the SQI specification in heterogeneous use cases.

3. SQI supports, both, a stateful and a stateless implementation.

4. SQI is based on a session management concept in order to separate authentication issues from query management.

7.5.2 Synchronous and Asynchronous Query Mode

SQI can be deployed in two different scenarios.

1. In the synchronous scenario (Figure 7.18), the target returns the query results to the source. Results retrieval is therefore initiated by the source through the submission of the query and through other methods allowing the source to access the query results.

2. In the asynchronous scenario (Figure 7.19), results retrieval is target-initiated. Whenever a significant amount of matching results is found, these results are forwarded to the source by the target. To support this communication the source must implement a results listener. The source must be able to uniquely identify a query sent to a particular target (even if the same query is sent to multiple targets). Otherwise the source is not able to distinguish the search results retrieved from various targets and/or queries previously submitted to a target.

Please note that the asynchronous query mode does not require an asynchronous handling on the messaging layer. It can also be implemented by two synchronous functions at the source and the target, respectively.

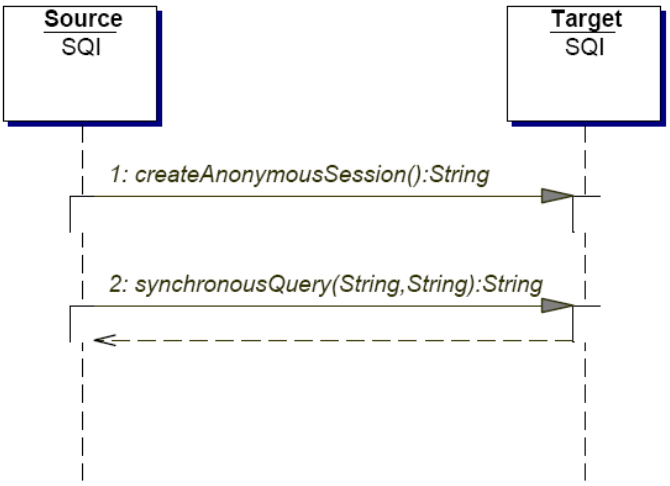


Figure 7.18. Synchronous Query Mode

A query interface operated in synchronous mode can perform multiple queries per session (even simultaneously).

In case of an asynchronously operated query interface, the source provides a query ID that allows it to link incoming results to a submitted query (the source might query many targets and each target might answer to a query by returning more than one result to the source). Multiple queries can also be active within a session in asynchronous query mode.

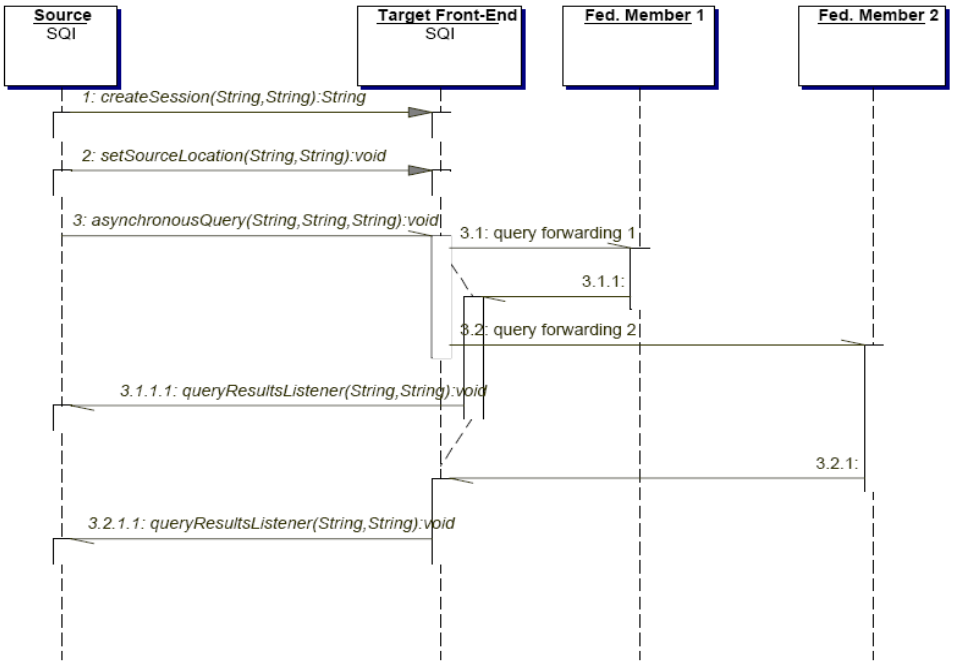


Figure 7.19. Asynchronous Query mode used for performing a federated search

7.5.3 The Simple Query Interface – An Interoperable Application Service for Querying

While for many aspects of the Interoperability Framework wide-spread solutions do exist, the educational domain is still lacking interoperable application services that take advantage of standards such as IEEE LOM, WSDL/SOAP, and XML. In this section we propose an application service for querying that can be used for federating heterogeneous learning object repositories.

An application service for querying needs to specify a number of methods a repository can make available in order to receive and answer queries from other applications. To distinguish the requestor from the answering system in our scenarios, the term 'source' is introduced in order to label a system which issues a search (the source of the query). The term 'target' labels the system which is queried (the target of the query). Alternatively, the 'source' can also be referred to as 'requestor' and 'target' as 'provider'.

Metadata can be stored using different means, such as file-based repositories, (possibly distributed) relational databases, XML databases, or RDF repositories. In order to make learning repositories interoperable, not only a common interface needs to be defined, but also a common query language together with a common results format for learning object descriptions needs to be agreed on.

The query service is used to send a query in the common query language to the target. Next, the query results, represented in the common results format, are transported to the source. On the implementation level, wrappers may need to be created to convert a query from a common query language X to a local query language Y and transform the query and the query results from a proprietary format to a common one and vice-versa.

Figure 7.20 illustrates an exchange process, where Learning Repository A (the source) submits a query to Learning Repository B (the target). It is assumed that both systems have agreed upon a common query language beforehand. The concepts used in the query statement are part of a common (query) schema. At Repository B, the interface component might need to transfer the query from the common query language to the local one. Also some mappings from the common to the proprietary schema might be required before submitting the search. This task is performed by a wrapper component. Once the search has yielded results, the results set is forwarded to the source, formatted according to a common results format.

The collaborative effort of combining highly heterogeneous repositories has led to the following requirements.

1. The application service needs to be neutral in terms of results format, query schema and query language. The repositories connecting can be of highly heterogeneous nature: therefore, no assumptions about these components of the interoperability framework can be made.

2. The application service needs to support synchronous and asynchronous queries in order to allow the application to be deployed in various use cases and heterogeneous network architectures.

3.The application service needs to support, both, a stateful and a stateless implementation.

4.The application service shall be based on a session management concept in order to separate authentication issues from query management, but also for providing an anchor point for implementing simple business models for access control.

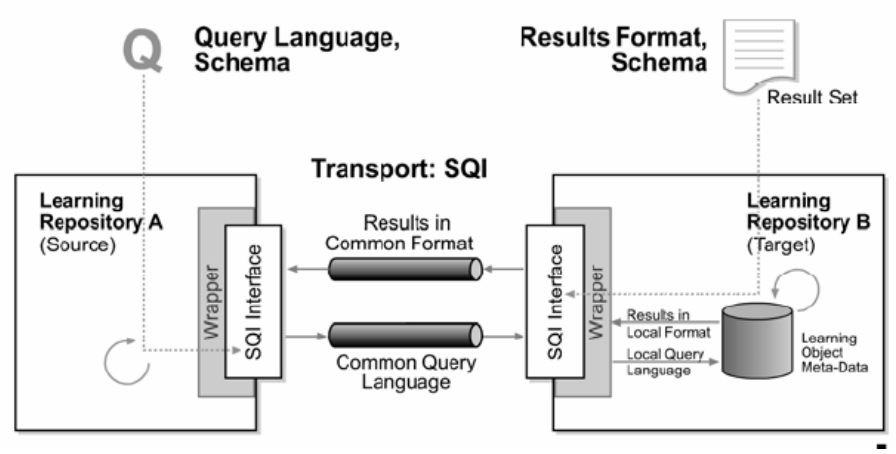


Figure 7.20: Communication between two Repositories

7.6 Five-year Project Plan

7.6.1 Phase I

1st Year	Background Study, Requirement Analysis and Feasibility Study
	1. **Background Study** 1. Literature Review 1. Smart Sensors 2. Smart Objects 3. Distributed Tuplespace 4. Ontology-based Modeling 5. Lightweight Database 6. Ontology-based Database 7. Profile-based Storage 8. Query Engine 2. Reviewing the existing system 1. Methods used 2. Performance 3. Identify scope of improvement 2. **Requirement Analysis** 1. Identify the Challenges to Build the System 2. Identify the Project Requirements based on the Challenges 1. Methods 2. Resources 3. Developers Skills 3. Feasibility Study

	1. Identify the End Users for the System 2. Identify Users Needs from the System 3. Availability of Technology to Meet the Requirements

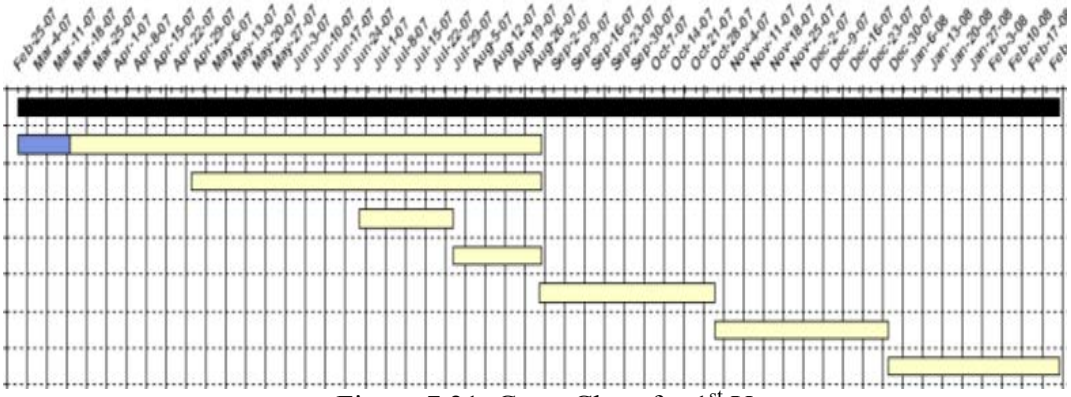


Figure 7.21. Gantt Chart for 1st Year

2nd Year	Architecture Specification and System Design
	1. Architecture Specification 1. Distributed Tuplespace Architecture 1. Local Tuplespace Architecture in Smart Objects 2. Associative Memory based Distributed Tuplespace Architecture 3. Identify interaction among Smart sensors and objects 2. Ontology-based, Lightweight Database Architecture 1. Concepts Definition 2. Lightweight Database Architecture 3. Agent Definition 3. Identify Collaboration among Smart Objects and Handhelds 1. Query Interface Architecture 2. Interface for Smart Objects using handhelds 4. Context Repository in Backend Server 1. Modeling repository 2. Modeling Context Linker and Merger 3. Query Interface 5. Finalize architecture specification 1. Integrating Components into same Platform 2. Finalize Overall System Block Diagram **2. System Design** 1. Design Each Component 1. Tuplespace Format Specification 2. Lightweight Database Tuple Format Specification 3. Backend Server Context Repository Schema Definition 2. Co-ordination among each Component 1. Design API's for Intra-Interaction 2. Design API's for Inter-Interaction 3. Software Agent Design 3. Incorporate query interface (e.g. SQI) 4. Finalize Overall System Design 5. Prepare Class Diagrams 6. Prepare Sequence Diagrams

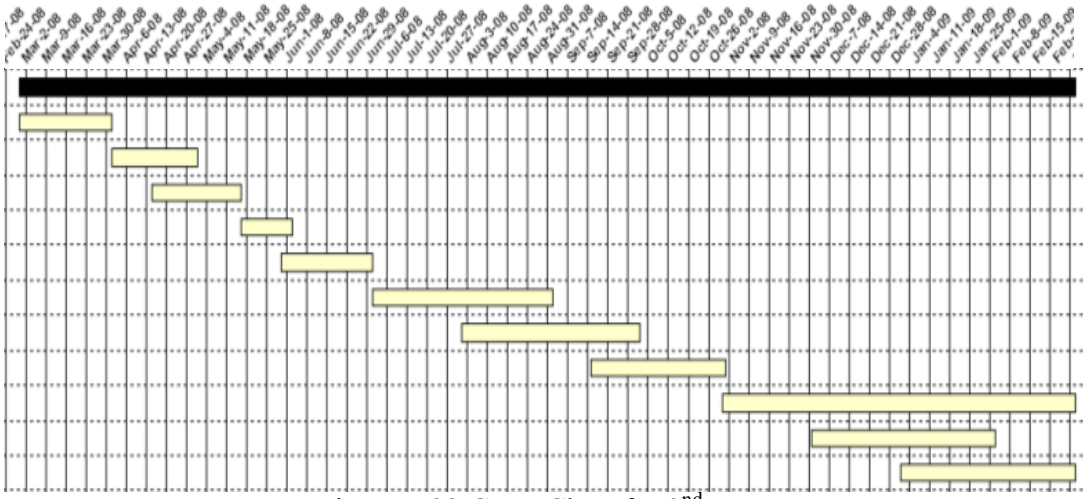


Figure 7.22 Gantt Chart for 2nd Year

3rd Year	Implementation
	1. **Implementation** 1. Ascertain Availability of Candidate Technologies 2. Choosing Technology and Platform to Implement 3. Implementing Each Component 1. Distributed Tuple Space 2. Ontology-based, Lightweight Database 3. Context Repository in Backend Server 4. Implement API's for Interaction among Each Component 1. Intra-Sensors Interaction 2. Smart Sensors- Smart Objects, Handhelds Interaction 3. Intra-Object Interaction 4. Smart Object- Handhelds Interaction 5. Smart Object- Server Interaction 6. Handhelds- Server Interaction 5. Provision for Reusability 6. Start Documentation

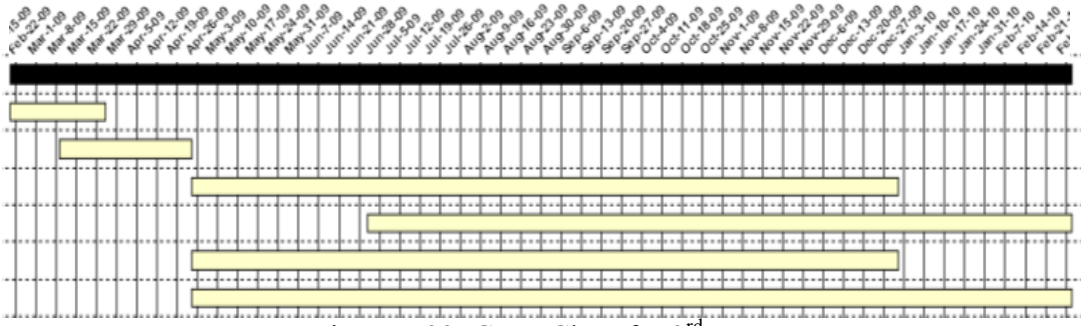


Figure 7.23: Gantt Chart for 3rd Year

7.6.2 Phase II

4th Year	Testing and Performance Evaluation

	1. **Testing**
	  1. Unit testing
	  2. Integration testing
	  3. Overall system testing
	2. **Performance Evaluation**
	  1. Evaluate the Performance
	  1. Each Component Performance
	  2. Overall System Performance
	  2. Find out the Performance Bottleneck
	  3. Changing Code for Increased Performance
	  4. Regression Testing

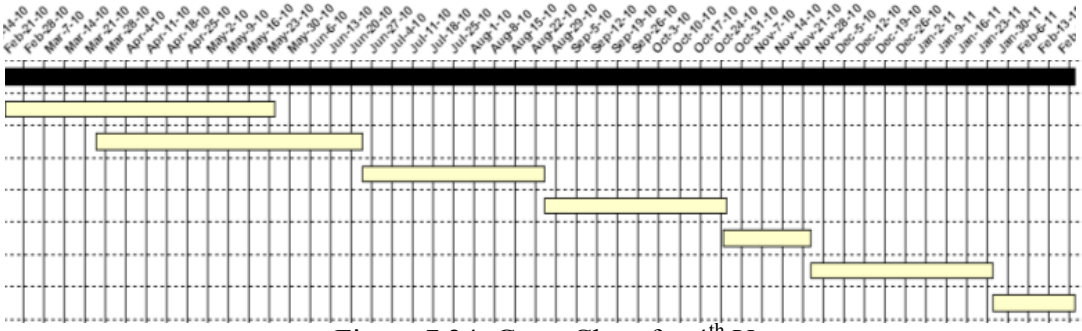


Figure 7.24: Gantt Chart for 4th Year

5th Year	Deployment and Support

	1. **Deployment**
	  1. Deploy the Developed System into Real Environment
	  2. Find out System Bugs
	  3. Fix the System Bugs and Re-Deploy the System
	2. **Support**
	  1. Prepare User Documentation
	  2. Train Users and Operators with the System
	  3. Finalize Developers Documentation
	  4. Proper Maintenance of the System

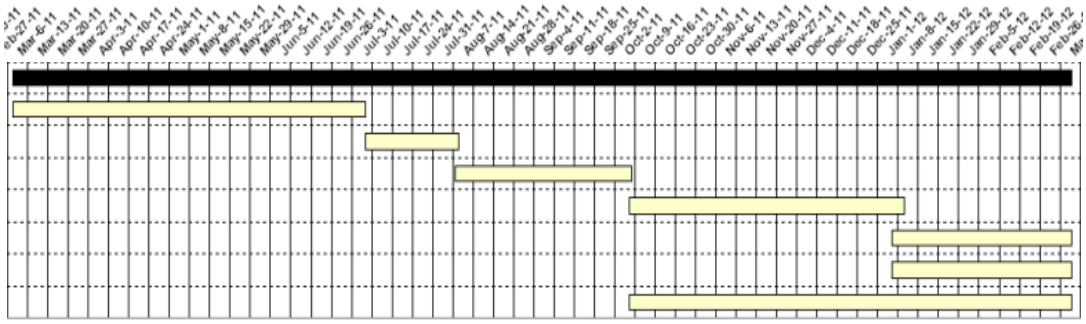


Figure 7.25: Gantt Chart for 5th Year

7.7 Conclusion

Though our proposed context repository scheme still in a abstract level of thinking, it can satisfy the requirements for SCO environment. We are going to use one of the light-weight databases that will take out a huge overhead of computational power and memory. Moreover, each entity will hold the ontology for itself that introduce the provision of distributed modular ontologies [30, 31]. Ontology provides a data model that can serve as a common framework for context storing thus make knowledge sharing and reuse effective. Logic inference is very much possible by using the Semantic Web Rule Language (SWRL) [32] and a rule engine like Java Expert System Shell (JESS) [33]. Our next step is to further investigate the strengths and weaknesses of the proposed architecture for context storing and once finalize the architecture then, to implement the context repository for SCO environment.

7.8 References

1. T. Gu, X. H. Wang, H. K. Pung, D. Q. Zhang. An Ontology-based Context Model in Intelligent Environments. In Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2004), pp. 270-275. San Diego, California, USA, January 2004

2.Thomas Strang, Claudia Linnhoff-Popien: A Context Modeling Survey. Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England, September 2004

3.W3C. Composite Capabilities / Preferences Profile (CC/PP). http://www.w3.org/Mobile/CCPP.

4.WAPFORUM. User Agent Profile (UAProf). http://www.wapforum.org.

5.BAUER, J. Identification and Modeling of Contexts for Different Information Scenarios in Air Traffic, Mar. 2003. Diplomarbeit.

6.HENRICKSEN, K., INDULSKA, J., AND RAKOTONIRAINY, A. Generating Context Management Infrastructure from High-Level Context Models. In Industrial Track Proceedings of the 4th International Conference on Mobile Data Management (MDM2003) (Melbourne/Australia, January 2003), pp. 1–6.

7.SCHMIDT, A., BEIGL, M., AND GELLERSEN, H.-W. There is more to context than location. Computers and Graphics 23, 6 (1999), 893–901.

8.CHEVERST, K., MITCHELL, K., AND DAVIES, N. Design of an object model for a context sensitive tourist GUIDE. Computers and Graphics 23, 6 (1999), 883–891.

9.MCCARTHY, J. Notes on formalizing contexts. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (San Mateo, California, 1993), R. Bajcsy, Ed., Morgan Kaufmann, pp. 555–560.

10. MCCARTHY, J., AND BUVAˇC. Formalizing context (expanded notes). In Working Papers of the AAAI Fall Symposium on Context in Knowledge Representation and Natural Language (Menlo Park, California, 1997), S. Buvaˇc and Ł. Iwa´nska, Eds., American Association for Artificial Intelligence, American Association for Artificial Intelligence, pp. 99–135.

11. Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL Web Ontology Language Reference. February 2004. W3C Recommendation, available at: http://www.w3.org/TR/owl-ref/.
12. J. Hendler and D. L. McGuinness. The DARPA Agent Markup Language. IEEE Intelligent Systems, 15(6):67–73, 2000.
13. D. Brickley and R.V. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. 2003. W3C Working Draft, work in progress, available at: http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.
14. R. Agrawal, A. Somani, and Y. Xu. "Storage and querying of e-commerce data" In Proc. VLDB'01, pages 149–158, 2001.
15. B.McBride. "Jena: Implementing the RDF model and syntax specification." In Proc. of the 2nd Intern. Workshop on the Semantic Web, 2001.
16. S. Harris and N. Gibbins. "3store: Efficient bulk RDF storage." In Proc. of the 1st Intern. Workshop on Practical and Scalable Semantic Systems (PSSS'03), 2003.
17. L.Ma, Z. Su, Y. Pan, L. Zhang, and T. Liu. "Rstar: an RDF storage and query system for enterprise resource management." thirteenth ACM international conference on Information and knowledge management, 2004:484 – 491.
18. S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, and K. Tolle. "On storing voluminous rdf descriptions: The case of web portal catalogs." In Proc. ofWebDB'01 (co-located with ACM SIGMOD'01), 2001.
19. J. Broekstra, A. Kampman, and F.V. Harmelen. "Sesame: A generic architecture for storing and querying rdf and RDF schema." In Proc. of the First Inter. Semantic Web Conf., pages 54–68, 2002.
20. Z. Pan and J. Heflin. "Dldb: Extending relational databases to support semantic web queries." ISWC'2003, 2003.
21. H. Dehainsala, G. Pierra, L. Bellatreche, "OntoDB : An ontology based database for data intensive applications", DASFAA 2007.
22. YounHee KIM, ByungGon KIM, and HaeChull LIM. "Design of Storage Structure for Path-Based Query on RDF and RDF Schema." IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 2006. Volume E89-A , Issue 6. pp. 1733-1735
23. Mohammad-Reza Tazari. "A Context-Oriented RDF Database. The first workshop on the Semantic Web and Databases." Tazari, M.R. Berlin (Germany), September 2003.
24. Wikipedia: http://en.wikipedia.org/wiki/Tuple_space.
25. M.H. Tsou, B.P Buttenfield,"An agent-based Communication Mechanism for distributing geographic information services on the Internet", GiScience 2000, Georgia.
26. SQLite: http://www.sqlite.org/
27. HSQL: http://hsqldb.org/
28. Mini SQL 2.0: http://www.hughes.com.au/library/msql/manual_20/
29. CEN/ISSS (2005). A Simple Query Interface Specification for Learning Repositories (CEN Workshop Agreement #15454). Brussels, Belgium, retrieved 23March 23, 2006.
30. Y.Qu, Z.Gao, "Interpreting Distributed Ontologies", WWW 2004, USA.
31. Jie Bao, V.Honavar,"Adapt OWL as a Modular Ontology language (a position paper)"

32. SWRL: A Semantic Web Rule Language Combining OWL and RuleML, http://www.w3.org/Submission/SWRL/

8. Knowledge Processing

8.1 Introduction

8.1.1 Ubiquitous Computing Vision

The term "Ubiquitous Computing" was originally introduced by Mark Weiser [1] in the year 1991. In his fundamental article *"The Computer for the 21st Century"* [2], he elaborated about "the computer that disappears". For Weiser the way into the 21st century was obvious: Computer and Network technologies are getting smaller, cheaper, and more powerful. Therefore, more and more everyday artifacts are going to be equipped with a reasonable amount of computing power and, maybe even more important, are networked together into a virtually unique network of communicating "things that think". In the pure sense of the word, computing gets "ubiquitous", anywhere, any time. Computers in every thing that is calmly doing what we intend it to do, in a way that is non-obtrusive and user-friendly, in a sense that we do not have to focus our attention on the trivia of running an electronic system.

Research on Ubiquitous Computing (Ubicomp) is related to very many other disciplines from Robotics and Embedded Systems, Networking and Distributed Systems, to

Artificial Intelligence and Psychology. Thus Ubiquitous computing is a very difficult integration of human factors, computer science, engineering, and social sciences.

8.1.2 Context-aware Computing

One goal of Context-aware Computing is to acquire and utilize information about the context of a device to provide services that are appropriate to the particular people, place, time, events, etc. For example, a cell phone will always vibrate and never beep in a concert, if the system can know the location of the cell phone and the concert schedule. Often, the term "Context-aware Computing" is used in a sense synonymously to Ubiquitous Computing. This is because almost every ubicomp application makes use of some kind of context. Ubicomp is mainly about building systems which are useful to users, which "...weave themselves into the fabric of everyday life until they are indistinguishable from it" [2].

For ubicomp systems, *Context* is essential. How can a system be helpful for a user? Users tend to move around often, doing new things, visiting new places, changing their mind suddenly, and changing their mood, too. Therefore, a helpful system seems to need some notion of *Context*.

In the Human point of view, we have a quite intuitive understanding of *Context*. Here, *Context* is often referred to as "implicit situational understanding." In social interactions *Context* is of great importance. A gesture, a laugh, or the tone of sentences builds up the implicit "picture" of what is meant or what communication partner is thinking. The same words can have a completely different meaning in different contexts.

In Computer Science, Context is quite a familiar concept, be it within the discipline of Artificial Intelligence ("Thinking machines"), in Robotics ("Adaptive Systems"), in User Interface Design (like adaptive UIs or office assistants like the Microsoft Office assistant called "Clippy"), or basically any other discipline (to some extent). Especially, every discipline dealing with human users tries to take into account human behavior one way or the other, with the generated output loops back as part of the vector of input values.

From the variety of definitions commonly used by Ubicomp researchers we can imagine how difficult it is to find a common ground. Context definitions are far away from mathematical precision and a particular definition often strongly depends on an authors' subjectiveness:

- Schilit and Theimer [3]: "Context is location, identities of nearby people and objects, and changes to those objects."

- A. Dey and Abowd [4]: *"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves."*

- Pascoe [5]: "Context is the subset of physical and conceptual states of interest to a particular entity."

So what is this leading to? Are those definitions helpful or misleading? In the sense of a functional definition they are only helpful as a general description of what to do. As an application designer they are only stating what they are doing anyway: trying to figure out what input is needed to produce the desired output. Hence, it is of topmost importance to have some common ground or a common "vocabulary" when talking about what *Context* is. We need some sort of *formal approach* towards handling and describing Context. Furthermore, in a software engineering sense, we need building-blocks for building context-aware applications in a structured way. The Context Toolkit [6] by A. Dey is a step into this direction and a good example for this principle (fig.1.) The Toolkit includes building blocks called "Widgets", wrapper classes for Sensors which serve as a hardware-abstraction layer, "Aggregators", which concentrate multiple input values to a single output value, and "Interpreters", implementing some application logic and generating application dependant "higher-level" output based on the input given. They interpret the incoming data according to a pre-programmed scheme.

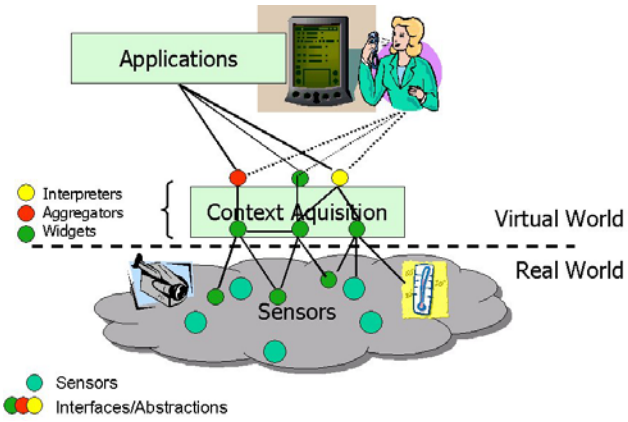


Figure 1: The Context Toolkit Core Components

With the Context Toolkit, the development of Context-aware applications basically consists of several distinguishable steps including

1) The real-world is sensed;

2) Context is detected, aggregated, "interpreted", and

3) Applications are custom-built to match the "context-detection" technology.

However, we believe that there is more tool-support necessary for software engineering and the design of Context-aware applications than provided today. We want to emphasize that the way applications are developed is very dependant on the underlying technology used, which we consider as bad practice in the long run. Research in the direction of

decoupling applications from data acquisition seems to be important. This is detailed in the section 2, *Middleware for Context-aware Ubiquitous Computing Environments.*

8.1.3 Role of Knowledge Processing

Formation of the context information relies heavily on the inter-dependency of the various events, entities and other domain objects. A formal approach for context formation should be able to recognize these inter-dependencies and combine the current information to form the effective context. This sort of an approach would heavily rely on domain-knowledge, the performance of a system made along these lines would also be affected by the knowledge representation scheme used in such a heavy knowledge-based system.

The Knowledge processing layer of CAMUS is mainly responsible for providing these knowledge-related services especially storing the domain information and definitions in an efficient manner, defining rules for context-processing and synthesis, providing adequate mechanisms for transforming the sensed data into useful complex concepts, providing efficient means of tracking user-location. These tasks can be considered as forming the fundamental aspects of the knowledge processing in a context-aware ubiquitous system. Additional functionalities can be defined to cater for more complex tasks which depend on the degree of interaction to be kept with the user and his environment. As the amount of interaction becomes more intense, so does the need to store more elaborate information, more complex mechanisms for context and complex-concept formation increases. In essence the Knowledge-Processing layer forms the brain of the system.

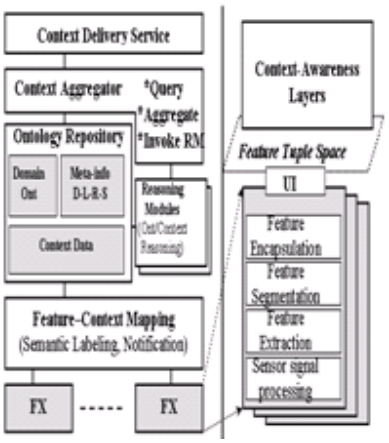


Figure-2: The CAMUS architecture

The current CAMUS architecture, as shown in figure-2, incorporates the knowledge processing components at all stages. The extraction of features and their conversion to context information forms the core operation of the knowledge processing layer. This task is achieved by the collaboration between the ontology repository, the feature-extraction layer, the reasoning modules. The reasoning modules range form general reasoning mechanism to specialized reasoning mechanisms such as the location reasoning module. The knowledge processing layer is responsible for carrying out all the needed tasks and then providing the outcome in the form of general or specific context

information to the service delivery layer which in turn is responsible for invoking the required services as mentioned by the application.

The knowledge processing layer requires very intensive computations for carrying out these tasks, it is needed to implement these knowledge-processing in a computationally efficient manner, thus there is a trade-off between the preciseness of the results and the speed with which the computations are performed.

8.2 Data Management in CAMUS

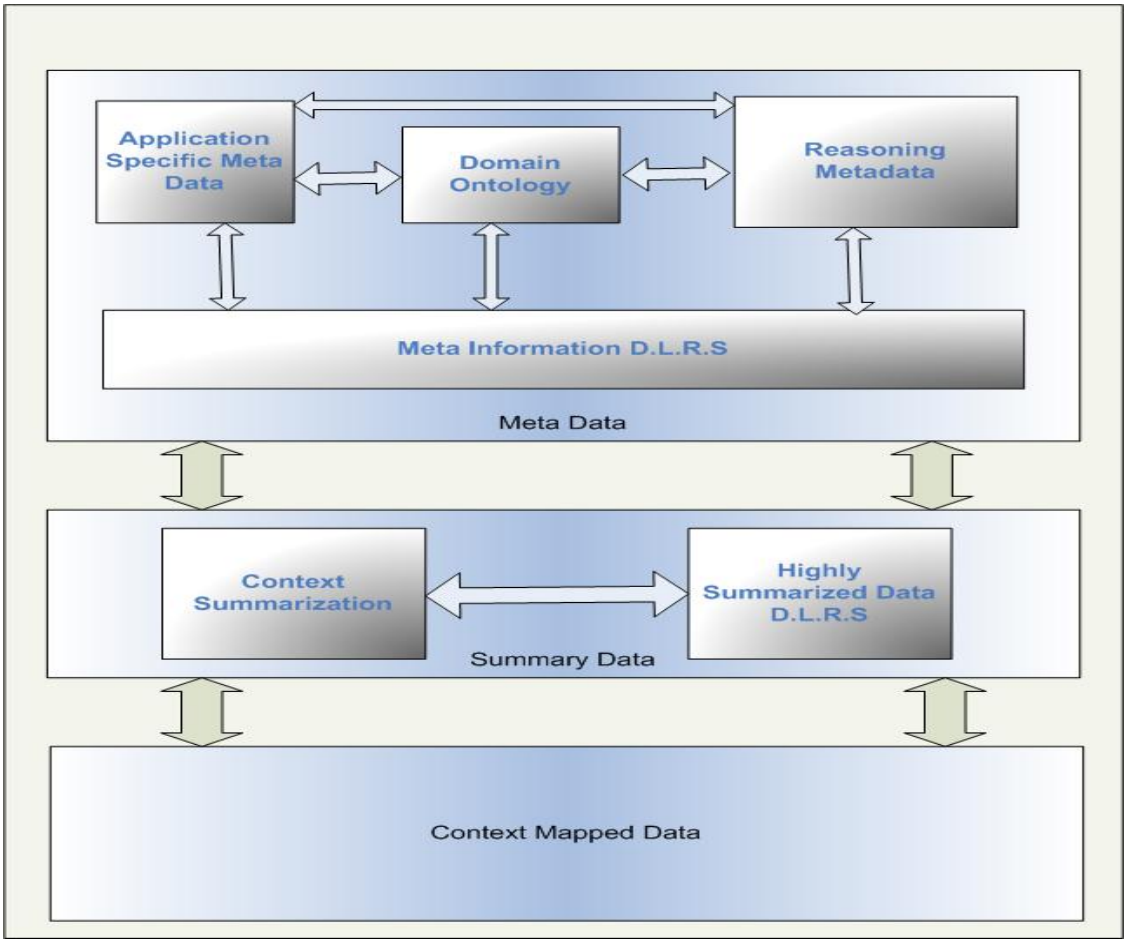
8.2.1 Motivation

Ubiquitous data management has a lot of challenges, which are not there in conventional data management application. The following list outline a few challenges, this list by no means complete. The ubiquitous application area is growing and a lot of challenges are still on the way.

- Users are moving from place to place (*Location Awareness*, *High Mobility*)
- Data are stored in many places. (*High Distribution*)
- Various devices are with different capabilities and they use different means to store/access data. (*High Heterogeneity*)
- Users cannot consistently control all the smart spaces he ever interacted with. (*High Autonomy*)
- Users generate certain data and may want to access others. (*Sharing and Collaboration*)
- The context of data is also important (*Context Awareness*)
- The source of data must be known (*Provenance*)
- Others: resource-constrained devices, unreliable connectivity and light weight communication and security measures.

CAMUS has a central repository to maintain data with context. Data is captures by lowest layer through different sensors and given to second layer which build context from that data using OWL Ontology. This context-mapped data is pertinent to query the context later on. The data repository is responsible to maintain this data for future uses. How would this repository attack all the challenges will be discussed later in this report?

8.2.2 Architecture



8.2.3 Meta Data Management

The Meta Data is very important in CAMUS as it gives an open way to accessibility and understanding of the data present in the repository. Metadata can provide richer searching and other services within a service and the glue for integration across several services. There are several key standard for maintaining metadata, we are using RDF based metadata to define and store data related to context and reasoning engines. The metadata is a separate storage in the repository that can be very helpful in migrating this metadata to some other application.

8.2.4 Data and Knowledge Management

The focus of data and knowledge management in CAMUS repository is 'doing the thing right' instead of 'doing things right'. It does cater to the critical issues related to adaptation, maintenance and manipulation of both data and knowledge present in the repository. In ubiquitous application the very nature of data required a lot of care and hence the data and knowledge management process and procedures and very important in this regards. One of the main issues in pervasive computing is that how to manage the context data over a large number of domains. A ubiquitous computing system can consist

of many subsystems running on various domains such as home domain, office domain, university domain, etc. Furthermore, many ubiquitous systems can collaborate with each other to build a large pervasive environment. The use of ontology can help sharing the knowledge about data among different domains and systems. However, such a distributed and dynamic environment requires an efficient mechanism to store and retrieve context data over multi-domain repository and at the same time we also required the metadata of this multi-domain to assimilate and understand different domain and context within these domain.

8.2.5 Knowledge Sharing and Querying

The knowledge sharing and querying process in CAMUS is very simple and elegant. Each access for querying and sharing requires going through the metadata interface and getting the required information access criteria from there, and then the actual access is being made depending upon the nature of query and information share.

8.2.6 Provision of Summary Data

CAMUS offers context summarization and garbage collection for the un-referenced and redundant context data. Similarly a variety of different reasoning engines are available in CAMUS to apply different type of reasoning required by any application, which is running on top of CAMUS. Summary data provide a lot of ease in information retrieval. Question arises here what to summarize and how to summarized the required data. The provision to keep summary data in the repository is there and the query processing exploits the fast that this summary data can decrease the query time.

8.2.7 Conclusion

In this chapter we discussed the role of repository in CAMUS, we are still evaluating different databases for performance reasons. There is no standard repository available to store and manipulate ontological data. The methodology we suggested here is a framework for knowledge management is ubiquitous environment. The selection of the repository is still on going project and will be incorporated in the next version of this technical report.

8.3 Context Summarization & Context Garbage Collection

8.3.1 Introduction

The idea of Ubiquitous Computing [1] is gaining the popularity with every passing day. Several research groups are developing their own ubiquitous computing projects [2] [3] [4] [5]. Ubiquitous (or pervasive) computing provides a computing environment where computing resources are spread through out, present everywhere in the environment and providing services to user seamlessly & invisibly without any explicit user intervention. A ubiquitous computing environment, thus, contains a number of devices, sensors, and software systems.

Context Awareness is among the foremost important features of any ubiquitous computing environment. In order to provide appropriate services to user, an application needs to be aware of the user and environmental context. Similarly at lower levels of abstraction, an application (or middleware) is also required to be aware of the computational context including device and network state. So what is 'context' itself? We take context as the 'implicit situational understanding' and consider all the information that defines a situation as context. So, location, temperature, network bandwidth, device profile, user identity can all be taken as the context information or simply context.

Since a Ubiquitous Computing system needs to deal with such huge and diversified information (context), there should be an appropriate context model to define, represent, and store the context efficiently in some context repository. The management of context information and data in ubiquitous computing imposes lots of issues and challenges. M. J. Franklin [6] has identified a number of such issues in ubiquitous data management such as those posed by adaptivity, ubiquity, mobility and context awareness

We approach the context (or data) management in ubiquitous computing from a different perspective. We are working on to identify the relevance and significance of information that a ubiquitous computing system receives from sensors and its surrounding. We believe that identifying and removing the irrelevant context (we call it 'garbage collecting context') and summarizing the available or incoming context (which we call the 'context summarization') will result in the improved performance of knowledge reasoning, inference making, machine learning and efficient use of computing resource including the storage space required by the Context Repository.

8.3.2 Problem Definition

Usually a ubiquitous computing environment comprises of a number of different sensors providing context information like

- Environmental context (temperature, pressure, light),
- Audio,
- Video,
- Location context,
- Computational context (network bandwidth, underlying operating system, hardware specification)

- The list goes on and on…

The context information comes in a continuous stream with each sensor emitting the data regularly (at least during some interested activity). We are heading towards flood of context data. Such a huge amount a data requires proper management and should be dealt with great care. At this point, we need to answer what to do with such a huge amount of data? Do we need to store all of this data? More importantly do we really need such a large amount of data?

Several data items sensed from the environment are required for some instant processing and reasoning, e.g., the presence of a person can be used to trigger the activity of turning lights on or caching the data related to the particular user. But most of the time, we also need to store the context for later use; knowledge reasoning, inference making and machine learning. For instance, we may need to keep the context of user presence for some on going (near future) activities, we may also need to store this information to reason about what she is doing and to make the inference what she might be up to. We also need this and other related context information (like time, and other activity details) for machine learning. Using various similar activities, a system can extract the patterns of activities and use such patterns to infer expected user intentions.

But storing all such context information imposes several issues. First, it requires considerable amount of storage space. Since ubiquitous computing systems are essentially distributed, therefore, migrating larger amount of data puts significant burden over network traffic. Secondly, the query processing and data retrieval on large context repository requires significant computing resources decreasing the overall throughput of the system. Thirdly, several contexts needs to be discarded and should not be stored permanently. For example, the data with low precision, because of noise, needs to be filtered out before sensitive operations (e.g., heartbeat rate of a patient). Privacy control also prevents us from storing each and every information, e.g., the information that user is in washroom. Lastly, the performance and efficiency of techniques such as knowledge reasoning, inference making and machine learning depends heavily on the size of supplied data.

So what can be done then? We present our proposed solution to cop up with such issues.

8.3.3 Proposed Solution

First we need to identify the low precision, irrelevant and redundant context and the one that is no longer useful and remove such context information. We call this process as Garbage Collecting Context (GCC).

Secondly, we need to summarize the actual (raw) context in such a way that it is more meaningful, can be used more efficiently for reasoning, etc and takes up less storage space. We name this process as the Context Summarization (CS).

A simple analogy is the human behavior towards the received news. Every day, we read a lot of news in newspaper, on internet and through television. But do we (need to) remember all the words and information that make up a particular activity or event? What we actually (need to) remember is some compact information about a particular event that what has actually happened. For example, Bob watches a soccer match for 70 minutes with a lot of attention but after the match is over, he does not remember exactly what had happened in the 14^{th} minute of the game. What he actually remember is some pattern or a summary of the match like who has won the match, few ups and down during the match and how many goals were scored and by whom. This is very close to what we mean by **Context Summarization** that instead of storing each and every raw information, only keep the summarized and meaningful context information. Coming back to the scenario, after the match is over, Bob tends to forget some information, for example, how far did the ball go when Player X kicked it and who received it. Also, as time goes by, he also tends to forget more details like a spectator had broken in to the game field. This act of discarding irrelevant information is analogous to the concept behind **Garbage Collecting Context**

In the following subsections, we will discuss, in more detail, about Garbage Collecting Context and Context Summarization.

8.3.3.1 Garbage Collecting Context (GCC)

Garbage Collecting Context is analogous to the concept of garbage collection in programming languages [7] [8] where we try to identify the memory areas no longer needed by a program and free it. Similarly in GCC, we try to identify the relevance and significance of context data and filter out (remove) the irrelevant, redundant and useless context.

Garbage Collecting Context can be used at various places. It can be used to filter out the noise in the data, i.e., the data with low precision so that it does not affect the efficiency of system actions taken on the basis of incoming context. For example, in some hospital system if a particular context value of the heartbeat rate of a patient is not sensed accurately because of some noise and interference, we should discard it before entering the system. Several systems [5] provide the precision value or the probability of the correctness of sensed value which can be employed.

Garbage Collecting Context (GCC) can also be used to identify and remove the context that is no longer needed by an application. For example, if in an application, we are getting and storing temperature values after every 5 minutes then an application may not require such context information for a large period of time. It might be useful to discard this history after certain period of time say 3 days. But generally, discarding information is not considered as a good idea; therefore, here we can employ the idea of context summarization and replace the raw history with this summarized history.

Privacy control can also be dealt using the Garbage Collecting Context. In this case, certain privacy policies determine which context should not be stored and included in the system processing and should be discarded. For example, the location of user in private

places (like washrooms) and activities during the lunch break should not be processed and stored permanently in the system. Similarly, the list of private telephone calls received and made by the employees (e.g., to or from family members) should not be preserved by the system.

Garbage Collecting Context (GCC) Manager is a strong candidate to deal with such issues and solve them efficiently.

8.3.3.2 Context Summarization (CS)

Where Garbage Collecting Context (GCC) identifies and removes the irrelevant and insignificant context, the Context Summarizer (CS) operates on the incoming and existing context data to
extract the useful information from the original data, and
convert existing context information to more useful form

so that the output context consumes less storage space and improve the performance and efficiency of query processing, data retrieval, reasoning mechanism and machine learning. The summarized context thus produced, replaces the existing raw context.

8.3.4 Context Summarization (CS)

Lets consider an example of context summarization. Consider a temperature sensor emitting the temperature value after every 5 minutes. We can simply store this as it is coming. Table 1 demonstrates this case.

Table 1. Temperature values stored after every 5 minute

Time	Temp.
12:05	23 °C
12:10	21 °C
…	
15:35	15 °C
…	

Using Context Summarization, for example, we can summarize this information and group on the daily basis. Table 2 demonstrates one such implementation

Table 2. Temperature values stored daily

Date	Avg. Temp	Max. Temp	Max. Temp At	Min. Temp	Min. Temp At
12/01	8 °C	14 °C	15:15	2 °C	04:35
12/02	7 °C	15 °C	14:55	0 °C	06:05
12/03	9 °C	13 °C	12:40	-1 °C	03:50
…					

Another possible implementation could be achieved when a day is divided into several periods like morning, afternoon, evening, early night and late night and context information is kept for each such period. Table 3 demonstrates such approach

Using the similar concept, location and computational environment context (like network bandwidth and processor load at a particular time) can also be summarized.

Table 3. **Temperature values stored for different periods of day**

Date	**Period**	**Avg. Temp**	**Max. Temp**	**Min. Temp**
12/01	Morning	5 °C	8 °C	3 °C
12/01	Afternoon	10 °C	14 °C	8 °C
12/01	Evening	9 °C	11 °C	7 °C
12/01	Early Night	7 °C	8 °C	5 °C
12/01	Late Night	4 °C	5 °C	2 °C
…				
12/04	Morning	4 °C	8 °C	1 °C
12/04	Afternoon	8 °C	11 °C	5 °C
…				

The above examples demonstrate the summarization on historical data, i.e., the data that has been recorded earlier. The context summarization can also be used as the data is received from the sensor. For example, when receiving the data from some audio and video sensors we can summarize it by extracting the useful information from it. With audio, we can extract information like Intensity, Spectral centroid, Transient detection, Low Energy ratio and Audio type (Music, Talk, Telephone Ring, etc). From video sensor, we can extract information like Pixel percent change, Pixel change variance, Motion pattern, Luminous intensity, etc. As a result, instead of storing the complete audio & video context, we can summarize it and only store the relevant information that what the particular audio and video data represent.

One of the benefits of performing the context summarization is reduced storage space. Such a compaction of data repository will save significant amount of storage space which will result in the faster query execution and data retrieval. It will also make the data migration in distributed environment more efficient as the larger the size of context data needs to be migrated the more will be the burden on network traffic. But this is not the only motivation for context summarization. After all, storing the data in Giga Bytes (GBs) is not much a problem in terms of storage space. The primary motivation behind the idea of Context Summarization (CS) is to store only the relevant context information in such a way that it is more useful for context consumers.

Reasoning about the context and drawing inferences based on the context is the primary reason why we are keeping the context in context repository in first place. Reasoning and inference making are the primary tools for providing context aware services to the user. For example, if a ubiquitous computing system knows that when Bob comes to his office

in the morning, he likes to check his emails, then a system can start downloading his emails when Bob enters the room in the morning. What makes a reasoning engine perform more efficiently is the amount of source data and the quality of supplied data. We believe that if the context summarization is done properly according to the nature of target context then it will result in less data; optimized for reasoning and inference making. Likewise, machine learning also depends heavily upon the historical data and if it is supplied with relevant, compact and optimized context data (by context summarization) then it is also expected to perform better and will provide efficient results

Context Summarization is a goal driven task, i.e., if we know why we are summarizing the data and for what purpose it will be used, then the method opted is likely to produce more efficient results. In short context summarization is the process of transformation of raw context data into summarized more useful, relevant context data. Figure 1 depicts this

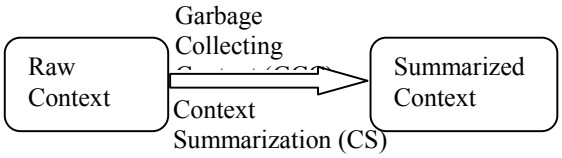


Figure 1. Context Summarization (CS) and Garbage Collecting Context (GCC) Process

8.3.4.1 Instantaneous & Delayed Summarization

Context Summarization can be classified into '**Instantaneous**' and '**Delayed**' in terms of when the summarization is applied. In **Instantaneous Context Summarization**, the context is summarized instantly as it is received from the context sources; sometimes, even before it being stored in the Context Repository. For example, the summarization of audio and video context (as discussed earlier in this section) comes in this category. Instantaneous Context Summarization is usually irregular and event-based and is performed more frequently.

Delayed Context Summarization is usually performed on the context already stored in the context repository. The summarization of temperature (as discussed earlier in this section), location, humidity and available network bandwidth comes in this category. Delayed Context Summarization, usually, is regular and periodic, i.e., performed in the background after a certain regular interval or at some pre-specified time. Delayed CS is usually performed less frequently and may consume considerable amount of computing resources

Figure 2 demonstrates the flow of context in Instantaneous & Delayed context summarization

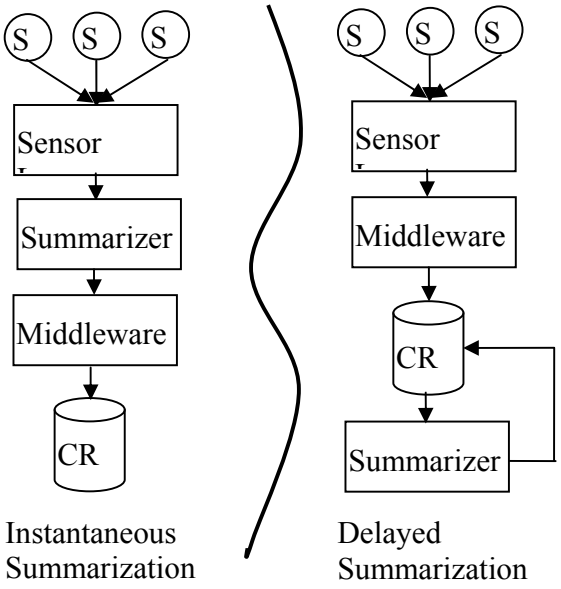


Figure 2. Instantaneous & Delayed Summarization

8.3.4.2 Three Levels of Summarization

Context Summarization can be performed at three levels. First of all, summarization can be applied at the **sensor layer** by summarizing the context information just as it is received from sensors. The summarization of audio/video context through feature extraction can be carried out efficiently at this level. Secondly, the summarization can be carried out at the **middleware level** when the context has been retrieved from the sensors and system environment and stored in the context repository. The summarization of numerical valued contexts (like temperature, pressure, available network bandwidth, etc) through aggregation and generalization techniques and that of location and activity information through pattern identification can be performed at this middleware level. Finally the summarization can also be carried out at the **application level**. Here the application specific logic specifies how to carry out the summarization and on which context information. The summarization of user and device profile through categorization technique falls under this level of summarization. Figure 3 depicts the three levels of context summarization

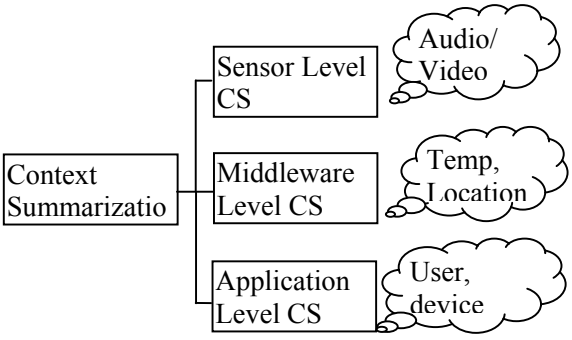


Figure 3. **Three Levels of Context Summarization**

8.3.4.3 Autonomic Context Summarization

Context Summarization is purely done for the ubiquitous systems internal use; hence it must be managed by the system with the least possible involvement of the human administrators. Our context summarization model [21] is policy driven where context summarization techniques exists as self managed components with each serving each different category of summarization. The administrator only specifies the policies through meta-data for each context type and the system configures itself accordingly. The system do not remove the context history just as it summarizes it but keep it in the secondary or backup storage and performs the lazy deletion as the history gets matured, thus providing system the ability of self healing which proves handy when something goes wrong. Thus our model justifies the self configurability, self management and self healing properties of IBM Autonomic Computing System model [22]

8.3.5 Techniques of Context Summarization

Now we will present various techniques that can be employed for Context Summarization in ubiquitous computing systems. We have identified several categories of context information based on the similarities in the context information and nature of context. Each technique is designed for a particular category of context and depends upon the nature of the context.

8.3.5.1 Aggregation

In aggregation, the history of context information is aggregated to generate compact and consolidated context. Numerical context types like temperature, light intensity, pressure, humidity, available network bandwidth and state of current system resources can be summarized using this technique. In previous section, we have demonstrated how this technique can be used to summarize numerical context information history.

Aggregation is a delayed, regular and periodic type of context summarization, which usually works in the background periodically after certain time interval or at some specific time and is usually less frequent. The Aggregation Context Summarization removes the original (raw) context after the context summarization has been performed.

Note that aggregation is performed when the context information has become the history and is not directly useful for application. Such historical information is useful mostly in identifying user preferences, machine learning and adaptation, reasoning and inference making.

8.3.5.2 Categorization

This technique of context summarization categorizes different context entities and summarizes the context values of these entities. For example, context information like user profile and device profile can be categorized to form user or device groups having some similar properties. In this way, we can identify the activities and features of a particular group or category like we can track the network bandwidth utilization by some particular user group (say doctors) or by some particular device group (say PDAs) during office hours.

Categorization is delayed and usually static type of context summarization, i.e., it is not performed and changed frequently. Categorization can be performed at system startup by some human or the system can learn itself and define categories as it is executed for elongated period of time. In any case, the categorization supports Machine Learning and higher level reasoning. Unlike other techniques, Categorization does (should) **not** remove the original context information such as existing user or device profiles.

8.3.5.2 Context Extraction

In Context Extraction, useful and interested context is extracted from continuous context streams such as audio and video streams. For example, Context Extraction can be applied to video stream received from video sensors like Camera, Webcam to extract features like pixel percent change, pixel change variance, picture motion pattern (such as stable, regular, irregular), luminous intensity, etc. In the similar way, audio context can also be summarized.

Context Extraction is an instantaneous, irregular and event based context summarization. It can start at any time whenever an interested activity starts. Unlike other techniques of context summarization, it can be triggered even before the context is stored in the context repository. In fact, it may discard the original (or raw) context even before it being stored in the repository; hence resulting in only storing the extracted features and not the original data. It results in saving a lot of storage space but may take considerable time in doing so. Some ubiquitous computing projects [16], including our project CAMUS [5], have been using this technique for some time.

8.3.5.3 Pattern Identification

Context information can be summarized by identifying existing patterns in the context repository or history of activities. For example, the location context can be summarized using this technique. Consider the location context history stored in the context repository as depicted in Table 4

Table 4. Location Context History of Users and Rooms

Time	User	Room
09:05	1	1
09:02	2	1
09:02	3	1
10:08	1	2
10:37	5	2
10:59	6	3
11:26	3	3
11:44	3	3
…		

Using pattern identification, a system may deduce the pattern of user's location during week days and come out with something as presented in Table 5.

Table 5. Pattern Identification for User Location

Time Period		User	Room	Probability
From	To			
09:00	12:00	1	1	0.76
13:00	17:00	1	1	0.83
09:00	12:00	2	2	0.67
13:00	17:00	2	1	0.89
14:00	19:00	4	3	0.36
…				

In the similar way, system can find the pattern of room occupants during various time periods. Using categorization along with pattern identification, system may also infer which user group (doctors, programmers, operators) occupies which room at different time periods.

Pattern Identification is again delayed, regular and periodic class of context summarization, i.e., it is invoked periodically after certain time interval or at some pre-specified time and works in background. It is resource intensive and thus, performed less frequently. On the positive side, it results in reducing considerable amount of storage space and also supports higher level inference making, machine learning and in predicting future intentions of a user or expected behavior of a device in the current situation. As mentioned earlier, pattern identification works on the existing history of context and replaces the larger history with patterns of activities.

8.3.5.4 Generalization

In generalization, we map various ranges of context values to a general higher level context. For example, we can map the raw temperature, network bandwidth and user movement speed to general concepts as presented in Table 6, 7 and 8 respectively.

Table 6. Generalization of Temp. Range

Temp. Range (°C)	Generalized Weather
20 ~ 30	Hot
10 ~ 19	Moderate
0 ~ 9	Cold

Table 7. Generalization of Network Bandwidth

Network Bandwidth Range Available	Generalized Network Traffic
1 mbps or more	Mostly free
500 kbps ~ 1 mbps	Moderately used
Less than 500 kbps	Busy

Table 8. Generalization of User Movement Speed

User Movement Speed (km/hr)	**Motion Pattern**
1 ~ 5	Walking
6 ~ 15	Running
16 or more	Rushing

The examples presented, until now, only contain the one to one mapping of the context values to higher level context (or concept) so where is the summarization in this technique? Actually generalization is an instantaneous kind of summarization performed between the sensor and middleware. This real time summarization only supplies the values to middleware when there is the difference of context general state. For example, suppose the generalization CS module informs the middleware that the current room temperature is moderate then it will only re-inform the middleware (and thus the system above) when the temperature general state is changed from moderate state (to cold, hot or other).

This type of summarization will definitely reduce the processing burden from the middleware in addition to saving the storage space used otherwise for keeping each and every sensed temperature value in the context repository. Generalization is mostly instantaneous kind of summarization working between the sensor layer and middleware. We can combine generalization technique with other techniques like Aggregation and Pattern Identification to further optimize the system performance.

8.3.5.5 Drift Calculation

This technique calculates the drift or the change of behavior of contextual activities from some fixed known points. For example, consider the network bandwidth of a particular server monitored regularly in a ubiquitous system. The server is supposed to have 1 Mbps network speed. The system hourly monitors the available speed. With drift calculation, we can calculate the drift rate of the network speed during peak hours, normal hours or the average drift from the assigned speed. The system, for instance, may find that during peak hours 5 pm to 10 pm the drift rate is 20% else the drift rate is less than 10%. Now we can keep only this drift rate into the context repository as inferred result from the network speed monitoring.

Similarly, drift calculation is also useful in identifying the user preference. The system can calculate how often the user behavior is different (drifts) from the system assumed behavior. For example, a system may calculate how often it makes appropriate decision for user's favorite TV show or other user intentions and keeps on recording the drift in the context repository.

Mostly drift calculation is delayed summarization and is performed at middleware level. It is especially useful for application adaptation for user preference and machine learning.

8.3.6 Related Work

Unfortunately, Garbage Collecting Context (GCC) and Context Summarization (CS) have not yet got the attention of researchers. One primary reason is that most of the ubiquitous computing systems are academic projects and are still in the phase of development. Not many systems have been deployed in real environment and actually used for elongated periods. The issues identified in this work come in front only to one's attention when the actual system is deployed and run for considerable time in real environments. Also, the focus of research community in ubiquitous computing is not towards the context data management, its techniques and issues in this field. Most of the research is still going on in finding ways to make ubiquitous computing operational in first place.

Several existing ubiquitous computing systems support features like noise filtering, privacy control, feature extraction [9] [10] [5] but we believe that using separate components for GCC and CS with clearly defining the responsibility of each component will produce better results; mainly because of the separation of concerns. Also the perception provided by GCC and CS attacks the issues in different and clearer way.

In Database Management Systems (DBMS), there are techniques that deal with similar problems. Data mining [11] and data ware housing [12] use the concept of histogram [13] and multidimensional views of database and work on the aggregate, consolidated data instead of raw data to support the higher level decision making and to identify the hidden patterns in the data. This can be considered as related to the idea of context summarization. Hence, when we extract underlying meaning from the context data, it can be considered as something like 'Context Mining' where we extract higher level context from the lower level context. Online Analytical Processing (OLAP) and data mining is not done on the actual data but on the historical, consolidated and aggregate data while we are performing the context summarization on the actual context. The goal of data mining and OLAP is somewhat similar but we want to transform the raw context to summarized form taking less storage space and provide improved and efficient reasoning and machine learning. Anyhow, the concepts explored in the field of data mining and OLAP are highly useful for the Context Summarization.

Researchers in DBMS have also analyzed the time series data streams for very large databases [14] [15]. Here, they analyze the data coming in continuous streams with time. They have proposed solutions on how to manage, represent and store the time series data streams. This is also highly related to the context summarization.

In traditional DBMS, the data is seldom deleted. But in our context summarizer, we do remove the raw context once it has been summarized and higher context have been extracted. We believe that we can afford to remove certain context data in ubiquitous computing environment and replace it with summarized information. Why? The answer lies in why, in first place, we are storing the context? We are storing the context and maintaining context history so as to reason on context, draw inferences from the context and make the machine learn. As we mentioned above that if the context is summarized properly, keeping the target usage in mind, the application can reason, infer and learn about the activities more efficiently as what they need is the history and consolidated data which we are providing as a result of context summarization.

8.3.7 Proposed Model for GCC and CS

In this section, we will present our proposed model for designing and developing Garbage Collecting Context (GCC) and Context Summarizer (CS) and provide few implementation guidelines.

The first question, while designing and developing the GCC and CS, is should these components be part of middleware or not? We believe that making these components part of a middleware will yield us the re-usability of design and code and the specific applications will not be required to re-write all the logic and code again.

We prefer designing these components (GCC and CS) as frameworks [17] [18] so that applications only need to provide the **hotspots** (areas of specification) for their specific needs. Hence, **Garbage Collecting Context** (**GCC**) can be developed in such a way that application specific techniques for Noise Filtering and Privacy Policies can be induced even while the application is operational. XML provides a good solution to specify which kind of data can be considered as garbage. For example, an application can specify, through XML, that from 1 pm to 2 pm, there is a lunch time at room X, so the location and other activities of users over there should not be monitored. The GCC contains some pre-specified noise filtering techniques and privacy policies while newer or updated policies can also be inducted in it exploiting the framework based design of GCC module. Figure 4 shows the proposed architecture of Garbage Collecting Context (GCC) module.

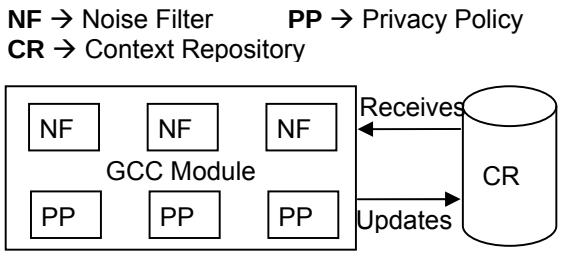


Figure 4. Garbage Collecting Context Module

The GCC retrieves context data from Context Repository (CR), identifies noise (corrupted) context using its Noise Filters (NF), applies Privacy Policies (PP) to remove

privacy sensitive context and updates the context repository. Depending on the implementation, GCC may not actually delete the context as it identifies the context as garbage but only mark that particular context information and later remove the context or move such context to some other repository for some human or system analysis.

Context Summarizer (**CS**) can also be developed with the framework technique. There are various context summarizer sub-modules for each different category of context. We call these sub-modules as Context Category Summarizer (CCS). Each CCS is responsible for dealing with each different category of context. Thus temperature, humidity, network bandwidth, luminous intensity can all be summarized using a single Context Category Summarizer (CCS). Context Summarizer (CS) is supplied context information along with Context Meta-Data (CMD). This context meta-data, usually represented through XML, specifies the type (or category) of supplied data, so that the CS may decide which Context Category Summarize (CCS) should be used to summarize this context information. All Context Category Summarizers (CCS) implement a particular interface so that the CS can access each of the CCS uniformly. Because of the framework based design of the CS, new CCS can be added and the existing CCS can be updated while the application is operational. Figure 5 shows the architecture of Context Summarizer (CS).

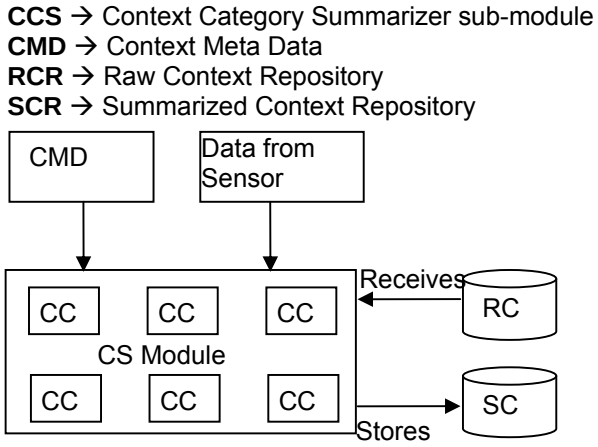


Figure 5. Context Summarization Module

Context is received from various context sources like sensors, computational and departmental infrastructure. The received context is then stored in some Raw Context Repository (RCR). Context Summarizer (CS) usually receives the source context data from this repository (RCR), summarizes and stores it in a separate repository called Summarized Context Repository (SCR). In practice, the RCR and SCR are not physically and logically different databases. Instead, they are managed in separate database tables, in case of related database management systems. Figure 6 presents the flow of Context in the presence of Context Summarizer.

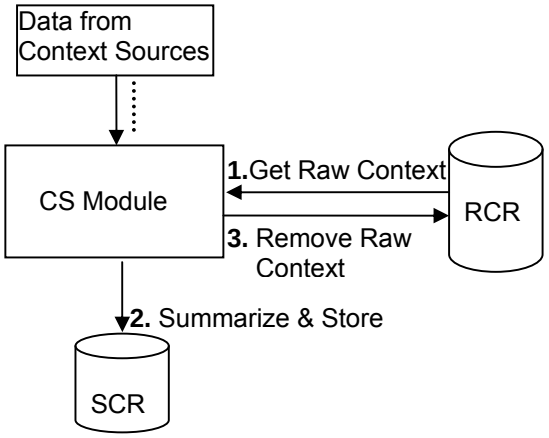


Figure 6. Context Flow in case of Context Summarizer

We are using our middleware CAMUS [5] to apply the context summarization. The interaction of summarization module with other components of middleware is presented in Figure 7.

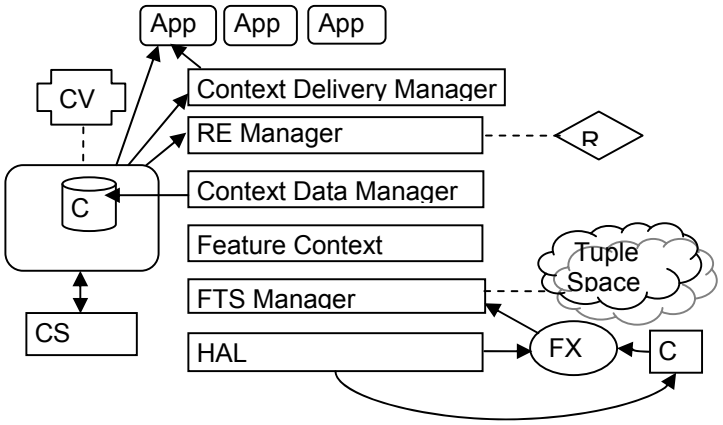


Figure 7. Interaction with other middleware modules

First we extract features (unified representation of sensory data) through our Feature Extraction Agents (FXA) and store all these features in Feature Tuple Space (FTS) which is an in memory repository of current context or the latest information received from sensors. As a new instance of information is inserted in FTS, the older one is transferred to the Context Repository (CR) represented using ontology in OWL through Feature-Context Mapping Layer. From then, all the middleware modules (reasoning engines, middleware services) and application access this information from the context repository. As data is stored in the repository, we summarize this information timely and store back to repository. One approach (used in case of temperature, humidity, etc) the raw information is removed from the repository and only the summaries or aggregates are used to answer queries. Another approach is to keep multiple summaries of different strength are kept and used to reply the query with appropriate confident values. A hybrid

approach can also be used in which both summaries and raw information are kept; specific or precise queries are answered from raw data while the general queries are answered through summarized information

Query Translation

Context Summarization modules change the context repository and form data units with different schema than the original one. How can context consumers cater with this? How do they know whether particular information is in summarized state or it is still in raw form? As in Figure 2, there is a special module called Query Translation (QT) which encapsulates context repository (CR). All other modules (CS, Reasoning Engine, Applications, etc) interact with repository through QT. Query Translator makes all the access to CR transparent, i.e., even the modules and applications are not required to be aware of summarization process. It keeps track of partition of summarized and raw data and directs the access to these accordingly by intercepting each and every access to CR. If the required data has been used in the summarization, it directs the queries to the summarized data repository. The results produced due to QT are not 100% accurate; hence it also returns a confidence value with each query result. Further, a query may also specify the minimum degree of confidence for the required results.

Context Category Summarizer (CCS)

Each category of context is summarized by a particular Context Category Summarizer (CCS); hence there is a different CCS for aggregation, pattern identification, etc based information. For example, temperature, available network bandwidth and noise level can be summarized using aggregation based CCS. Each CCS instance contains

(a) summarization algorithm,
(b) general parameters (key field, required fields, etc),
(c) specific parameters (source & target data source, summarization strength, time interval for repeated invocation of summarization),
(d) query translator for summarized information

Context Summarization Manager also maintains a list of context information used by different CCS for summarization, an example for such a table is present in Table 9.

Table 9. List of context information summarized by different CCS

CCS_ ID	CCS_ Instance_ID	Context_Type _ID	Last_Updated
017	1	1 (temperature)	09/19/05 05:42
017	2	4 (light intensity)	09/19/05 11:37
019	1	21 (location_A)	09/19/05 17:16

Using this list, a Query Translation Manager can identify whether a particular context information type is summarized and also if the required data has been in summarized or it

is still in raw format. Moreover, if the required information is in summarized state, then which CCS's QT should be invoked to get the query result? The general process flow of query processing is presented in Figure 8

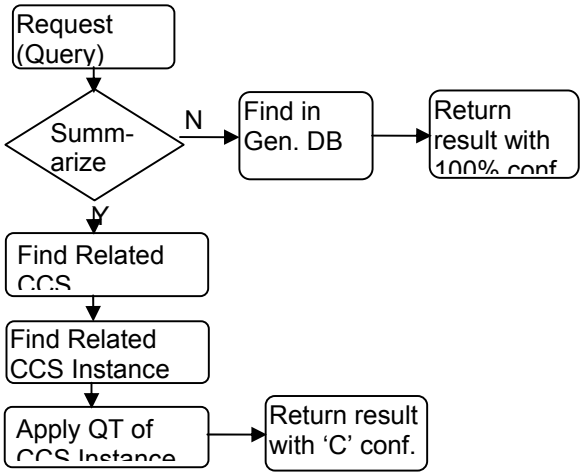


Figure 8. General Process Flow of Query Processing

Issues & Challenges

Context Summarization (CS) has its own unique research issues and challenges both at conceptual and implementation level. The issues range from questions like what/when/how context can be summarized to the performance and security of the ubiquitous computing system. In the following subsections, we will identify several such issues and wherever possible identify few applicable solutions.

Performance Overhead

Perhaps the foremost concern to apply Context Summarization (CS) techniques is the performance cost. What will be the performance requirement and what will it provide in return? Do the benefits achieved by these methods justify the computing resource consumption? We believe that a proper application of CS (like those discussed in section 4) will yield the performance improvement and will not eat up many resources. In any case, the overall system should not be ceased or hung-up during the execution of CS modules, the resources (like context repository) should not be locked for noticeable period of time and the regular execution of the system should continue without any disturbance or interruption by these modules. But the problem is how to achieve this? We need Context Summarization only when there is considerable amount of context information; a considerable amount of context means a considerable amount of processing and resource consumption to produce useful output. Designing algorithms and techniques to minimize this resource consumption is probably the biggest issue in Context Summarization.

Security & Risks

Security is the most questionable part of today's computing systems. What security threats would the techniques of context summarization will pose? The CS modules operate totally inside the middleware and ubiquitous systems and directly access and modify the context information which is the most valued asset of any ubiquitous system. Hence, the components and modules must be administered and validated carefully.

About the risks involved, firstly Context Summarization (CS) results in some data and precision loss. Failing to compensate this precision lost may result in decreasing the performance and overall throughput of the system. Secondly, improper Context Summarization may make the reasoning and machine learning even more difficult, complicated, inefficient, incorrect and misleading instead of improving it. Finally, CS makes changes to the existing Context Repository (CR). Several modules of middleware and application might be accessing the CR at the same time. Such a sudden modification might be unexpected for these modules and may make them produce unexpected results and must be avoided.

Incorporation of Summarized Context

Incorporation of summarized context into existing context repository is another research issue. The important point to note here is that the summarization process is performed when the ubiquitous system is completely operational. The context information might be in use when summarization modules access it to summarize and attempt to remove some information replacing them with the summarized context. But how the application processing can be diverted to the summarized context in between the processing? Our idea is to direct all the access to context repository first to the summarized part of repository. If the system requirement can not be fulfilled by it only then the access should be forwarded to the un-summarized part of the repository. The access to un-summarized repository should block all the CS modules on this part of un-summarized repository until the request has been fulfilled. To avoid such collision, we also recommend making the summarization on context history as periodic and scheduled according to the system load.

Other Issues

Some other research issues and challenges are;

- Using ontology for context representation is gaining acceptance in ubiquitous computing community [5] [20]. We need to define & implement techniques of CS that can efficiently operate on the ontology based context repository [19]
- How are we going to deal with the distributed and ubiquitous nature of middleware, data repository and applications?
- What are the security, trust and service level guarantees required for systems using CS techniques?
- What could be the possible impacts of these techniques, especially when something goes wrong?

In order to make Context Summarization (CS) feasible, we need to solve these issues and answer the challenges posed by these concepts.

Risks Involved

Garbage Collecting Context (GCC) and Context Summarization (CS) are sensitive in nature as they directly access context information and modify it. Information is always one of the most important assets of any system and organization. Hence, techniques like these must be applied with great care. In this section, we will briefly mention about some risk factors that should be considered while developing and implementing GCC and CS techniques

Garbage Collecting Context (GCC) and Context Summarization (CS) both will result in some data and precision loss. Failing to compensate this precision lost may result in decreasing the performance and overall throughput of the system.

Improper Context Summarization may make the reasoning and machine learning even more difficult, complicated, inefficient, incorrect and misleading instead of improving it

GCC and CS will make changes to the existing Context Repository (CR). Several modules of middleware and application might be accessing the CR at the same time. Such a sudden modification may be unexpected for these modules and may make them produce unexpected results and it must be avoided.

Future Work & Conclusion

The foremost important issue is the performance cost and the selection of time interval for the invocation of summarization. Synchronization of the different CCS modules is also an important consideration. If too many CCS modules start performing summarization then the overall system performance might degrade. Also there might be some queries for the data that is currently being summarized; we are also working on implementing the appropriate locking mechanism. Another interesting future work is to implement hierarchical summarization with different summarization strength and which allows inter-module negotiation [23] for required summarization strength and confidence values for queries. For our future work, we also want to use the concept present in [23] for summarization strength negotiation.

In the conclusion, we will say that Garbage Collecting Context (GCC) and Context Summarization (CS) are new, interesting and useful research areas and include a number of interesting research issues. We have presented both the benefits that can be achieved and risk factors that are involved in using these techniques and have also identified four different techniques for implementing Context Summarization (CS). We have also presented our proposed model for implementing these concepts and identified certain research issues and challenges we expect to face. We have concluded that these are sensitive operations and must be handled with great care and applied after rigorous testing. Finally, ‘to summarize and how to summarize?’ that is the question!

References

M. Weiser, The computer for the 21st century. *ACM SIGMOBILE 1999 Review*

Dey, A.K., et al.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Anchor article of *a special issue on Context-Aware Computing, Human-Computer Interaction (HCI) Journal*, Vol. 16. (2001)

Chen Harry, Tim Finin, and Anupam Joshi: An Intelligent Broker for Context-Aware Systems. In: *Ubicomp 2003*, Seattle, Washington

Gaia: A Middleware Infrastructure to Enable Active Spaces. Manuel Román et al., In *IEEE Pervasive Computing*, Oct-Dec 2002

Hung Q. Ngo, Anjum Shehzad, Saad Liaquat, Maria Riaz, Sungyoung Lee: Developing Context-Aware Ubiquitous Computing Systems with a Unified Middleware Framework. *EUC 2004*: 672-681

Michael J. Franklin, Challenges in Ubiquitous Data Management. . Informatics: 10 Years Back, 10 Years Ahead, LNCS #2000, R. Wilhiem (ed)., Springer-Verlag 2001

Henry Lieberman , Carl Hewitt, A real-time garbage collector based on the lifetimes of objects, *Communications of the ACM*, v.26 n.6, p.419-429, June 1983

Richard Jones, The Garbage Collection page, http://www.cs.ukc.ac.uk/people/staff/rej/gc.html

Mike Spreitzer, Marvin Theimer, Providing location information in a ubiquitous computing environment, *ACM SIGOPS Operating Systems Review , Proceedings of the fourteenth ACM symposium on Operating systems principles* Dec 1993, Volume 27 Issue 5

Jason I. Hong, James A. Landay, Support for location: An architecture for privacy-sensitive ubiquitous computing, *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, June 2004

Alex Berson , Stephen J. Smith, Data Warehousing, Data Mining, and OLAP, McGraw-Hill, Inc., New York, NY, 1997

Inmon, W.H., Building the Data Warehouse. John Wiley, 1992

D. Barbara et al., The New Jersey Data Reduction Report, Bulletin of the *IEEE Technical Committee on Data Engineering* December 1997 Vol. 20

Lin Qiao et al, Data streams and time-series: RHist: adaptive summarization over continuous data streams, *Proceedings of the eleventh international conference on Information and knowledge management*, Nov 2002

Approximating a Data Stream for Querying and Estimation: Algorithms and Performance Evaluation, *Proceedings of the 18th International Conference on Data Engineering* (ICDE'02), Feb 2002

Moore, D., I. Essa, and M. Hayes, Exploiting Human Actions and Object Context for Recognition Tasks, In Proceedings of IEEE International Conference on Computer Vision 1999 (ICCV'99), Corfu, Greece, March 1999

Mohamed Fayad, Douglas C. Schmidt, Object-Oriented Application Frameworks, *Communications of the ACM*, Volume 40 Issue 10, Oct 1997

Erich Gamma , Richard Helm , Ralph Johnson , John Vlissides, Design patterns: elements of reusable object-oriented software, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1995

Jena – A Semantic Web Framework for Java http://jena.sourceforge.net/

Anand Ranganathan, Roy H. Campbell, An infrastructure for context-awareness based on first order logic, *ACM Personal and Ubiquitous Computing*, Volume 7 Issue 6, Dec 2003
Faraz Rasheed, Y.K. Lee, S.Y. Lee. Context Summarization & Garbage Collecting Context: In Proceedings (II) of the International Conference on Computational Science & its Application 2005 (ICCSA 2005) (Singapore).
The Vision of Autonomic Computing. IEEE Computer, January 2003
Khedr, M. Karmouch, A: Negotiating context information in context aware systems. *IEEE Intelligent Systems* Dec 2004

8.4 Bayesian Reasoning in CAMUS

8.4.1 Introduction to Bayesian Reasoning

Bayesian reasoning is based on the celebrated Bayesian rule of conditional probabilities. Application of the Bayes' rule in very complex cause-effect maps requires intense computational and memory resources. Bayesian Networks were developed keeping this limitation in mind. Bayesian networks efficiently represent the cause and effect relationships which exist between the various domain features and at the same time they also provide adequate mechanisms for belief updation based on evidences extracted from the domain. Traditional reasoning mechanism like rule-based reasoning do not incorporate provision for uncertainty, Bayesian Networks have been designed so that reasoning can be performed under uncertain conditions.

A Bayesian Network consists of variable-set *V* (discrete/continuous random variables), a directed acyclic graph *G* and a joint probability distribution *P* defined over all the variables in the set *V* [1,2]. A sample Bayesian Network is shown in figure-1.

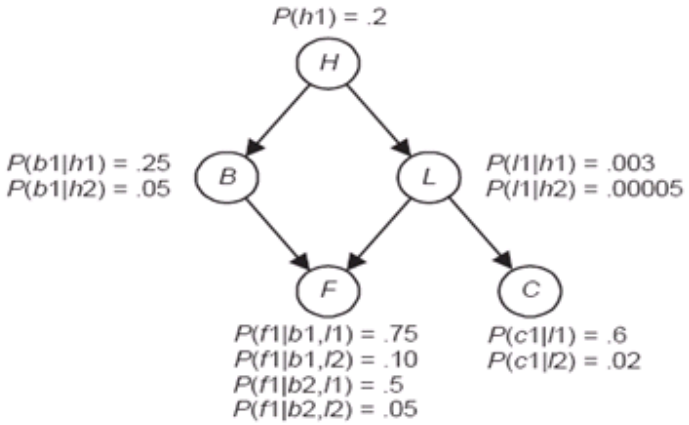


Figure-1: A sample Bayesian Network along with the probability distributions.

As the number of variables in a network grow the size of the joint probability distribution also grows exponentially. Thus it is better to store the conditional probability of each variable conditioned on its parent-set locally, as shown in fig-1. These locally stored probabilities can be combined using the chain rule [1,2] to construct the overall joint probability distribution *P*.

Bayesian Networks allow a structured representation of all the related concepts in a domain along with adequate representation of the strength of this relationship. The main language used for representing this strength is the probability calculus and the main motivation for using probability calculus comes from the well-formed axiomatic foundations of probability theory and the ease with which related beliefs can be combined [2].

The inference mechanisms for Bayesian Networks are also very well established and a number of inference algorithms exist which can be used to perform inference on sparse graphs. Algorithms for performing both exact and approximate reasoning exist and can be used depending on the domain being modeled and the requirements. Along with these algorithms for inference generation Bayesian Networks can also be learned from data, and similarly they also provide adequate mechanisms of adaptation and tuning. Adaptation can be viewed as online learning, and tuning is parameter adjustment (probability measure) in a supervised manner.

Bayesian Networks have been used in expert systems, decision support systems, fault and anomaly detection for industrial processes, automated planning etc.

8.4.2 Motivation

In a ubiquitous environment the system is supposed to infer high-level concepts from sensor and device readings. These sensor and device readings provide only very specific data about certain environment entities such as temperature, pressure, etc. As the system is unable to determine the exact nature of the concept (with complete confidence) the environment can be labeled as being partially-observable. For such partially-observable environments employing a technique which does not cater for uncertainty would be inappropriate.

As an example of the above mentioned description of a ubiquitous environment, consider the case in which the system needs to infer whether the user is having lunch or not. For inferring such an activity it is needed that we have some data about the location of the user, time of the day, and some data about his actions. Let's assume that the system has sensors for estimating the location of the user, similarly there are some pressure sensors on the chair in the dining room. If now a snapshot of the environment is taken into consideration and we find that the chair is indeed occupied, and also that the location of the user is the dining room. These pieces of information are not enough to correctly state whether the user is having lunch or not. It could be that he is reading some thing, and that he is just sitting there talking over the phone. Thus the system is now faced with multiple hypothesis (candidate conclusions) and it is needed that one of them should be adjudged as the most likely one. One way to break this tie among candidate conclusions is to give a score to each of these conclusions based on experience. This would require some data about the habits of the user, where and when he takes his lunch, how often he talks on the phone while in the dining room, does he usually reads in the dining room while not eating anything, thus the system requires some sort of probabilistic knowledge to rate these candidate conclusions. Once these conclusions have been rated the conclusion having the highest score can be considered as the most plausible conclusion which can be reached given the data at hand. Thus all that a system can do is behave in a normative manner, and predict these events in an uncertain manner. A Bayesian network for the example is shown in figure-2.

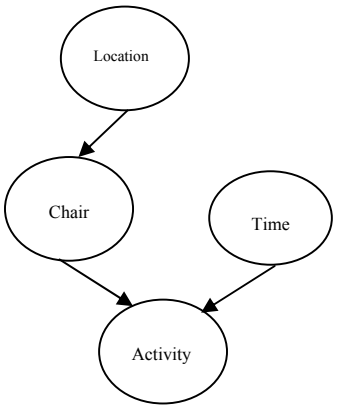
8.4.3 Where to use Bayesian Networks in A Ubiquitous Environment

Bayesian Networks are used for reasoning under uncertainty and should only be used for carrying out those tasks which involve a very complex structure of reasoning. They

should not be used for a task which can be solved without reasoning for example turning on the lights when motion has been detected. These kind of trivial tasks do not require much knowledge and it would be very costly as far as performance costs are concerned to use a Bayesian network for such a trivial task.

In Ubiquitous environments Bayesian Networks are best suited for accomplishing more complex tasks such as activity recognition, conflict resolution in a multi-user environment, assessing and modeling situations and advising the best action to take.

The design of a Bayesian Network is not an easy job, it requires that variables be identified and adequately modeled and then probability distributions for the root nodes be specified *a priori*. This requirement of prior measures can be fulfilled either through experience (a subjective guess by the designer) or they must be extracted from a large database of cases (objective measure based on recorded cases). Thus the most difficult task in the design of a Bayesian network is the specification of these probability measures. As this task requires a lot of skill and time it is not possible to model all the functionalities of a ubiquitous system through Bayesian networks hence it is needed that they be used only where complex decision making is required.



Every node represents a discrete random variable and each link contains the conditional probability,

Figure 2: A candidate Bayesian Network for the example.

8.4.4 Bayesian Reasoning in CAMUS

Context reasoning provided by the knowledge processing layer of CAMUS, has been designed to accommodate multiple reasoning paradigms, for reasoning about uncertain contexts [4]. The reasoning modules can be developed separately and then plugged into the context-reasoning layer.

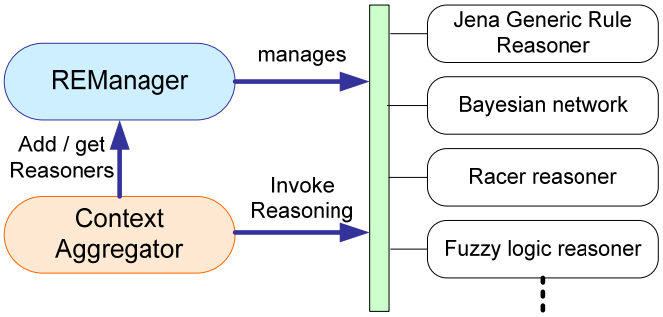


Figure 3: The design of CAMUS reasoning layer.

As can be seen from figure-3 that CAMUS provides a Reasoning Manager for managing multiple reasoning modules and a context-aggregator for invoking the reasoning processes and also managing the storage and retrieval of the high-level context so formed.

Another aspect to be considered is the knowledge-representation used in the context-repository of CAMUS, currently CAMUS uses OWL for representing ontologies about domain entities. Every reasoning paradigm has its own knowledge-representation so there is a need for translating the native knowledge-representation of the system into a representation which can be understood by the individual module. Similar translation is also needed for storing the high-level context formed through reasoning into the native knowledge-representation scheme of the system.

8.4.4.1 Design of the core Bayesian reasoning module

The main Bayesian reasoning module consists provides the basic operations for loading a Bayesian Network, absorbing evidences from the domain and providing the inferences on the basis of these evidences.

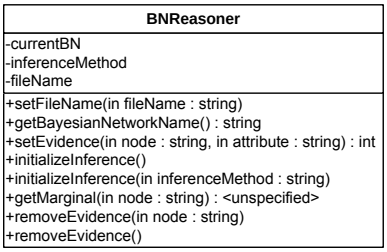


Figure 4: The core Bayesian Reasoning Module

The core module is able to handle a single Bayesian at any time instance. The network is designed and stored in an XML format. Other functionalities provided by the reasoning module is the entry and removal of evidences at any node in the network.
Various Inference mechanisms such as Pearl's Message Passing Algorithm [1], Junction Tree Algorithm for belief propagation [2], and the Bucket Elimination algorithm [2,3]. The output of the inference algorithm is the marginal probability distribution of each

variable. This marginal probability distribution represents the posterior odds (the belief in the variables' state) corresponding to the evidence entered and the prior distributions.

8.4.4.2 The Bayesian Reasoning Manager Module

As the core reasoning module is able to manage only a single Bayesian network, the task of the manager module is to spawn new instances of the core module, so that multiple networks can be loaded into the memory and concurrent reasoning can be managed. The Bayesian reasoning manager module receives requests from applications, and spawns new instances of the core module in the form of JINI™ services. These services are then provided the necessary input such as the XML files for the network structure and the probability distributions. A sequence diagram which shows the complete working of the Bayesian Reasoning process in CAMUS is given in figure 5.

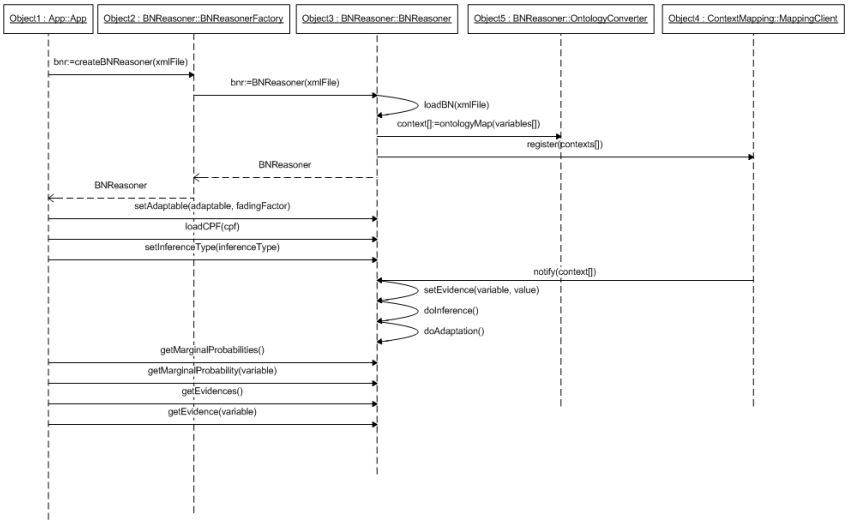


Figure 5: Sequence diagram for the Bayesian Reasoning Process.

The application can interact with the spawned reasoning service directly, and ask for any high-level inference which it wants.

8.4.4.3 Translation Modules

As mentioned previously there is a need for translating the knowledge maintained in CAMUS in the form of ontologies to a form that is required for creating a Bayesian Network. As an example consider the following piece of OWL:

```
<owl:Class rdf:ID="InDoor_loc">
        <rdfs:subClassOf>
                <owl:Class rdf:ID="Location"/>
        </rdfs:subClassOf>
</owl:Class>
```

Figure 6: Description of indoor location

The above piece of description logic defines the ‘indoors’ as a sub-class of location. One way of modeling this piece of knowledge in a Bayesian Network would be to define a discrete random variable named location, and represent the “indoor” as a state of this variable.
Similarly the context information is also represented as an ontology (in OWL) hence it is also needed that the higher-level inference generated by the reasoning module should also be converted back into the OWL format. Hence translation is needed twice in the reasoning process, once when the reasoning is to be initialized and again when the reasoning results are to be provided back to the application.
This translation also defines a mapping for the incoming data from the sensors and devices registered with the middleware to the variables in the network. Thus a mapping client is needed which can map this incoming data directly to the network nodes and instantaneous evidence absorption and propagation can be performed in the Bayesian Network.

8.5 User Preference Learning

8.5.1 Introduction

The most famous application of Context-aware computing are smart environments, such as smart home, smart office, smart campus, etc. Absorbing and developing the key idea of ubiquitous computing as well as context-awareness, a smart environment contains a large number of invisible sensors and actuators which enable the system to "think and work" base on its perception of user's context.

The logic to go from input to output of a smart system or from the sensor data to context and then to control commands depends on each system, each user and each kind of service. However, because the most important purpose of a smart system is to satisfy a user, every decision will be based on the user's preference. Therefore learning user preference becomes the most important task.

Here we need a formal concept for user preference. User preference can be understood as what the user wants the system to do in certain situation. For example, when Alice enters the house in a hot sunny day, she wants the air conditioner to be started 10 minutes before that, maintain the temperature of 20oC, and the curtain should be closed, while in rainy days she likes to open the curtain to enjoy the rain. But as for Bob, he wants the curtain to be closed on rainy days, and opened in sunny days.

The task of learning user preference has become very difficult due to many issues. First, the preference of user does not static. It changes quickly by time, which makes online learning (or adaptation) a crucial requirement.

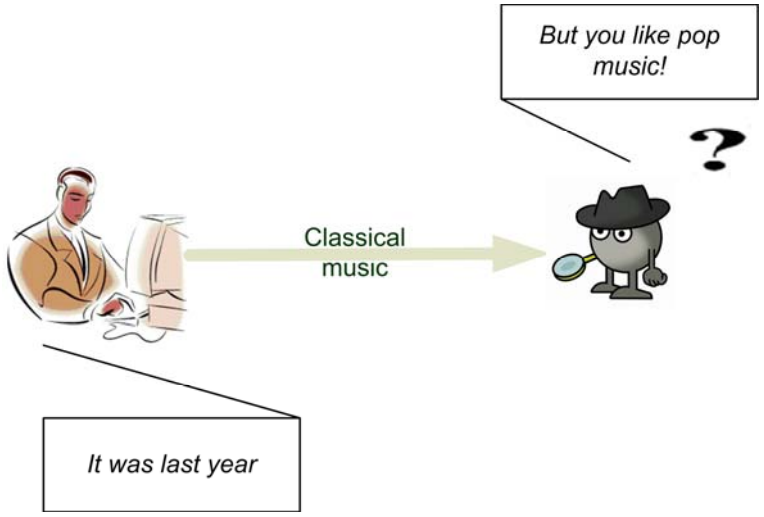


Figure 1: Problem when lack of adaptation.

Second is the uncertainty, which comes not only from the inaccuracy of sensor data but also from the ambiguity of users about their own preference.

Third, when there are many users in the smart environment, the desire of one user can be affected by others. It raises the challenges of distinguishing the preference of each user as well as resolving the conflicts among different user preferences.

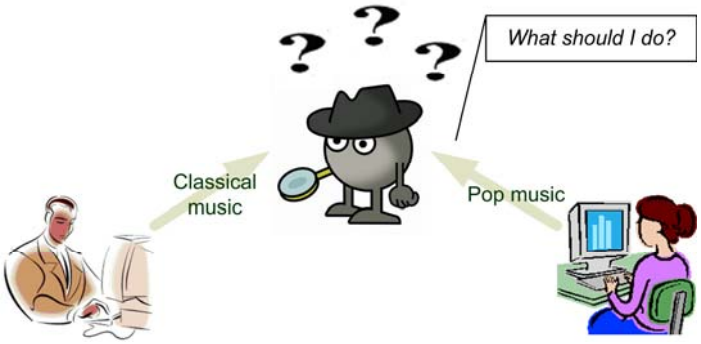


Figure 2: Conflict among user preferences.

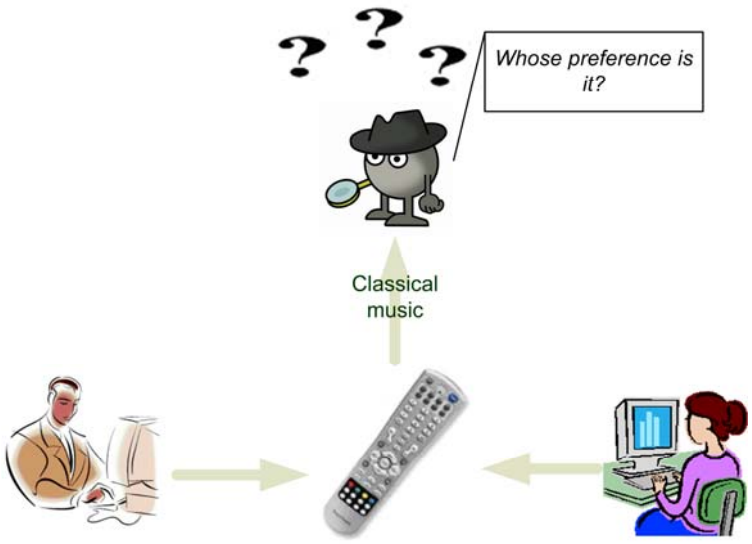


Figure 3: Distinguish among user preferences.

To deal with uncertainty in ubiquitous environment, Bayesian network [7] and Bayes theorem are widely used. Martin Muhlenbrock et.al. [8] uses a Bayesian approach to infer the notions of activity and availability from labeled sensor data in an office environment. The approach is rather simple without considering the multi-user activities. In another ubiquitous system project, to resolve conflicts for Context-aware Media services in smart home environments, GIST [9] proposes Conflict Manager which applies Bayes theorem. In order to resolve conflicts among users, the Conflict Manager sums preferences of users who are collided with each other and recommends the specific contents ordered by the summed preference. It also resolves conflicts among Media services by selecting a Media service with the highest priority. Furthermore, Conflict Manager resolves conflicts among Media services occupied to users by recommending the Media services. However, this approach lacks of a comprehensive model for user preference.

The Bayesian Metanetwork is first proposed by Vagan Terziyan et.al. [10], to select the appropriate substructure from the basic network level based on contextual features from user's profile (e.g. user's location). Two models of the Metanetwork are considered: C-Metanetwork for managing conditional dependencies and R-Metanetwork for modeling feature selection. An example of this approach is shown in Fig. 4.

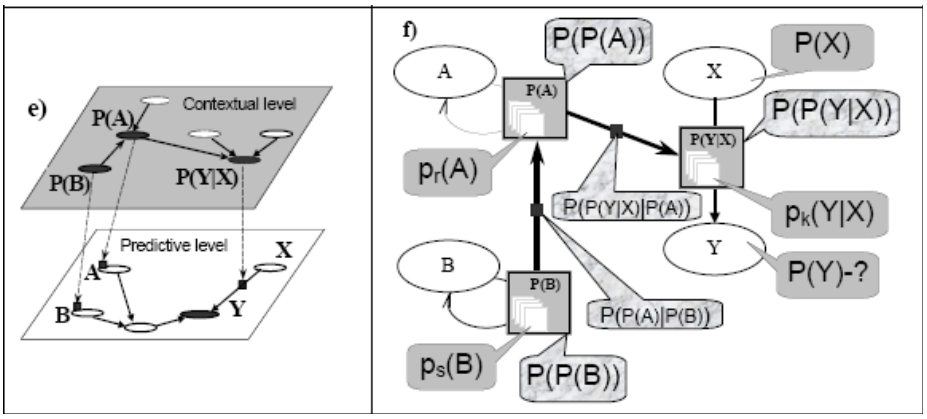


Figure 4: An combination example of R-Metanetwork and C-Metanetwork

However, neither adaptation algorithms for Bayesian Metanetwork nor detailed application for these models are mentioned. One meta-levels of C-Metanetwork is merely described as second order conditional dependencies among probabilistic distributions in previous level (i.e. all the nodes are conditional dependencies), and the R-Metanetwork only model the relevant features selection in a "Nodes-exist-or-not" manner. The big drawback of this approach is that the selection of relevance features and the distribution of conditional dependencies are modeled separately in two different Metanetworks, while in real world the change in a Bayesian network structure often leads to changes in conditional dependencies.

The limitations of current approaches raise the need of a method to learn user preference in ubiquitous environment which can satisfy following requirements:

- Dealing with uncertainty;
- Learning correctly and efficiently the preference of each user in a Multi-user environment;
- Reusable;
- Utilization of the domain knowledge and user-defined rules;
- Online adaptation to the newest user preference.

Besides, we see that when combining the preferences of many users, we often base on user priority. Priority can be fixed, but normally it changes by time, or by situation; for example a sleeping user will be considered more important than others. Hence the context-aware system should be able to learn and adapt to the changes of user's situation-based priority.

To address these issues, we propose a Bayesian RN-Metanetwork (RN stands for Relevant Network). A Bayesian RN-Metanetwork consists of many levels. In each level, there are many sets of Bayesian networks; the distribution of each set depends on the local probability distributions associated with the nodes of the next level network. we use the Bayesian RN-Metanetwork to learn both user preference and priority at the same time, as well as to resolve the conflict among many user preferences.

My approach differs from the previous ones in at least four aspects.

- First of all, by handling the relevant Bayesian network selection, the Bayesian RN-Metanetwork models can cover both the relevant feature selection and conditional dependency distribution at the same time. Its computations are simpler than those of C-Metanetwork and R-Metanetwork, but it can be used very efficiently for learning in multi-user or multi-agent systems. It also inherits the propagation algorithms of traditional Bayesian network, why in case of C-Metanetwork and R-Metanetwork proposal, the old propagation algorithms can not be applied, and they also have not given any algorithm for it.

- Second, the Bayesian RN-Metanetwork model is very suitable for multi-agent systems. Especially in ubiquitous environments, due to the complex, diverse and open-ended characteristic of the system, the multi-agent paradigm brings much more advantages than the single-agent paradigm.

- Third, the adaptation algorithm for Bayesian RN-Metanetwork is fully described to adapt the model to the continuously changing preference of users.

- Finally, no matter how many users are there in the environment, the priority and preference of each user is calculated separately. This is very useful for widening the scale of systems, as well as knowledge reuse.

8.5.2 Traditional Bayesian network

Probabilistic graphical models are graphs in which nodes represent random variables, and the (lack of) arcs represent conditional independence assumptions. Hence they provide a compact representation of joint probability distributions. Directed graphical models also called Bayesian Networks or Belief Networks [7].

In addition to the graph structure, it is necessary to specify the parameters of the model. For a directed model, we must specify the Conditional Probability Distribution at each node. If the variables are discrete, this can be represented as a table (CPT), which lists the probability that the child node takes on each of its different values for each combination of values of its parents. Consider the following example, in which all nodes are binary, i.e., have two possible values, which we will denote by T (true) and F (false).

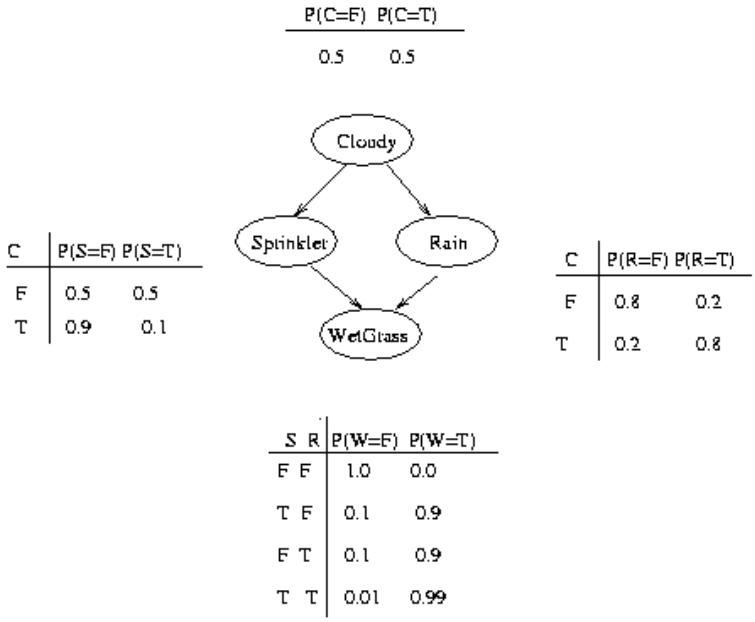


Figure 5: Example of Bayesian network.

Bayesian network is widely used in probabilistic modeling and dealing with uncertainty in machine learning. Some ubiquitous systems are using Bayesian network to learn the user activity, availability, etc. However, this "traditional" approach is restricted to simple use cases such as single user activity or single user preference. The next session explains why we need to replace the traditional Bayesian network by a Metanetwork, by discussing an example of multimedia service preference learning in a Smart Home system. It also introduces the concept of Bayesian RN-Metanetwork.

8.5.3 The Drawback of Traditional Bayesian network in Preference Learning

Let consider an example of using traditional Bayesian network to learn user preference about multimedia service. we assume that the multimedia service preference of user depends on user location, current activity, time and is different from user to user. If there is only one user in a system, this user's preference can be modeled by a basic Bayesian network as depicted in figure 6.

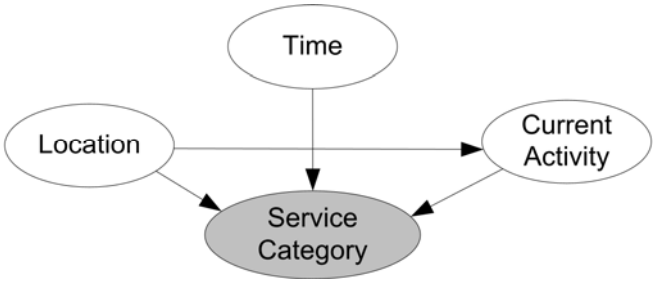


Figure 6: A simple Bayesian network to learn multimedia service preference of one single user. User's current activity is also related to user location.

In this case, it will be very easy for the system to learn the preference of user based on user's commands to select services. Location is acquired from location sensors such as RFID or wireless LAN (PDA), and current activity can be inferred using another Bayesian network [8]. Then the conditional probabilities in this model will be calculated using Bayes theorem.

However, when many users are present at the same time, there will be conflicts among the preferences. When a service is selected, the system can not know whose preference that service is. If there are three users in a smart home, one can think about another model to learn the user preference which is depicted in Fig. 7.

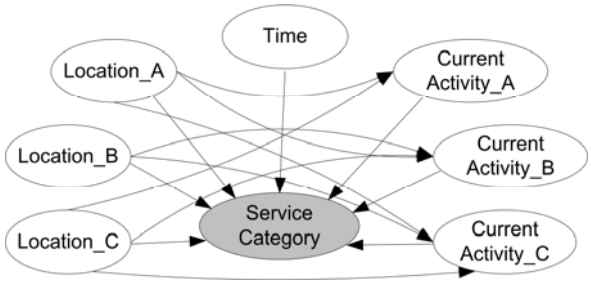


Figure 7: A Bayesian network to learn multimedia service preference of 3 users A, B, C. Each user's activity can be affected by others' location (in case of group activities)

Although a model likes the one in Fig. 7 is easy to create, the size and complexity of the network increases exponentially with the number of users and number of features which affect the preference. Moreover, using this model the system can not learn the preference of each user. In next sub-sections we introduce our approach using Bayesian RN-Metanetwork.

8.5.4 Bayesian RN-Metanetwork

The Bayesian network model can be divided into many sub-models, each of which models the preference of one single user with no affection from others. Then a Bayesian RN-Metanetwork will be used to manage the distribution of those sub-models and combine them in calculating the final preference.

Definition of Bayesian RN-Metanetwork: The Bayesian RN-Metanetwork is a set of Bayesian networks, which are put on two levels in such a way that the distribution of probabilistic networks on first level depends on the local probability distributions associated with the nodes of the second level network.

The Bayesian RN-Metanetwork is a triplet:

$$RMBN = (BN0, BNS, R)$$

where BNS = {BNS1, BNS2, ... BNSn} is a set of sets of Bayesian networks in first layer and BN0 is the second level Bayesian networks; R = {R1 ... Rn} is a set of interlevel links. The probability distribution of each Bayesian network is included inside it. Each Ri is a link "vertex – network set" meaning that stochastic values of vertex vi in the network BN0 correspond to the distribution of one set of Bayesian networks in the first level.

Bayesian RN-Metanetwork supports multi-agent systems. As depicted in Fig. 8, each set of Bayesian networks in the first level is hold by an agent. Each agent uses the distribution of its Bayesian networks to calculate some needed values, and they communicate with other agents through some interfaces. The interfaces consist of common nodes between agents' networks. The mechanism for belief updating in a multi-agent Bayesian network system is described in [29].

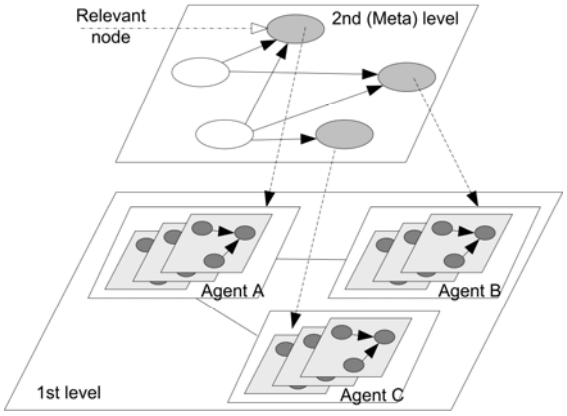


Figure 8: A Bayesian RN-Metanetwork structure.

The RN-Metanetwork can be freely expanded because any Bayesian network included can be itself another Metanetwork.

8.5.4.1 Modeling User Preference and Priority using Bayesian RN-Metanetwork

When many users are present in a smart environment, and each user has his or her own preference about a certain service, the last decision to select the service is related to the priorities of the users. For example, grandparents have higher priority than their grandchildren, so when the grandfather and his grandson are present in a smart home, the room temperature is adjusted based on the grandfather's temperature preference rather than that of the grandson.

Therefore, when modeling the user preference, we also need to model the user priority.

The user priority can be categorized into 2 types: situation-independent priority and situation-de-pendent priority. Situation-independent priority means that the priority of a user does not change when situation changes, i.e. it does not change by time or by place or under any condition. In contrast, situation-dependent priority changes when situation changes, such as when user changes his place or activity. For example, normally the

father has higher priority than his son, but when the son is sleeping, the audio volume and light are adjusted based on the son's preference at that time (audio OFF and light OFF).

The user preference in case of situation-independent priority can be modeled using the 2-level Bayesian RN-Metanetwork. Fig. 9 shows the Bayesian RN-Metanetwork for modeling user preference about Multimedia service in a 2-user system.

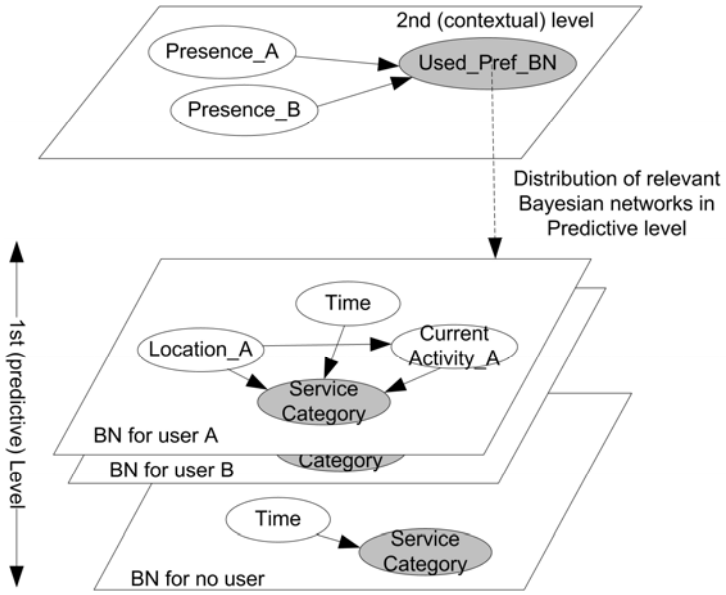


Figure 9: A Bayesian RN-Metanetwork to learn multimedia service preference of 2 users A, B in case of situation-independent priority.

The model is based on following idea: The priority of one user can be understood as how much that user can contribute into the final decision. In the model, the distribution of relevant Bayesian networks which model individual user preferences indicates the proportion of each user preference in the compound preference. Hence the meta-level of this Bayesian RN-Metanetwork also models the user priority. We can see that the user priority in this case depends only on the user presence. Table 1 shows an example of value of the conditional distribution of relevant Bayesian networks (or user priority).

Table 1
Conditional probability of Used_BN node in 2nd level Bayesian network in Fig. 4.

Presence_A	Y		N	
Presence_B	Y	N	Y	N
Use_A_pref	0.5	1	0	0
Use_B_pref	0.5	0	1	0
Use_Nouser_pref	0	0	0	1

From the conditional probability table, we see that the two users have same priority. When both users are present, each of them contributes 50% into the final preference decision.

In case of situation-dependent priority, the 2nd (or meta) level is replaced by a metanetwork. In other words, one more level for learning priority based on situation is added. Fig. 15 depicts the Bayesian RN-Metanetwork for the multimedia services preference in situation-dependent priority case.

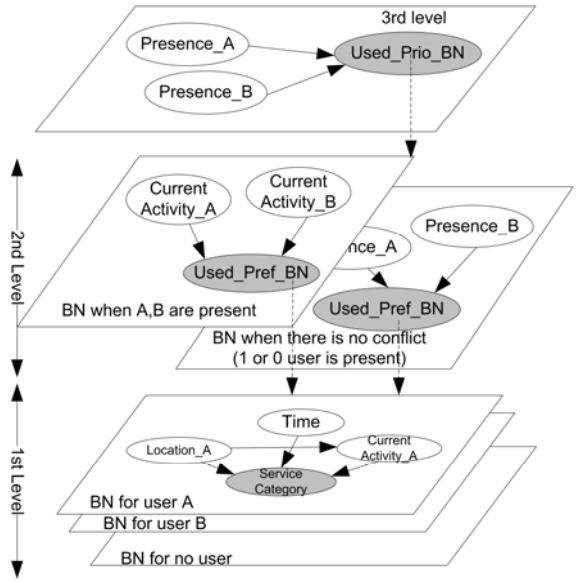


Figure 10: A Bayesian RN-Metanetwork to learn multimedia service preference of 2 users A, B in case of situation-dependent priority. Priority depends on not only user presence but also user activity.

Tables 2, 3 and 4 show the sample conditional probability of Used_Prio_BN node and Used_Pref_BN nodes. Note that the value range of Current activity nodes is reduced for easier demonstration (normally we have a lot of activities such as Reading, Walking, Eating, Working, etc.).

Table 2
Conditional probability of Used_Prio_BN node in 3rd level Bayesian network in Fig. 5.

Presence_A	Y		N	
Presence_B	Y	N	Y	N
Use_AB_prio	1	0	0	0
Use_single_prio	0	1	1	1

Table 3
Conditional probability of Used_Pref_BN node when both A and B are present.

CurrentActivity_A	Working		Sleeping	
CurrentActivity_B	Working	Sleeping	Working	Sleeping
Use_A_pref	0.5	0.3	0.9	0.5
Use_B_pref	0.5	0.7	0.1	0.5
Use_Nouser_pref	0	0	0	0

Table 4.
Conditional probability of Used_Pref_BN node in 2nd level Bayesian network when there is no conflict.

Presence_A	Y		N	
Presence_B	Y	N	Y	N
Use_A_pref	-	1	0	0
Use_B_pref	-	0	1	0
Use_Nouser_pref	-	0	0	1

When the priority model is simple and we don't want to make it redundantly complicated, we can use a traditional Bayesian network for the priority model, instead of using 2 layers of Bayesian RN-Metanetwork. The traditional Bayesian network for priority model in Fig. 10 is illustrated in Fig. 11.

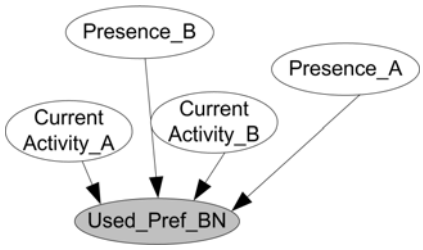


Figure 11: Simple Bayesian network for situation-dependent user priority

The algorithms for calculating the combined preference and learning user preference and priority are explained in next sections.

8.5.5 Probability Propagation and Adaptation for 2-layer Bayesian RN-Metanetwork

In this session, we will explain the algorithms for Probability Propagation and Adaptation for the basic 2-layer Bayesian RN-Metanetwork. The algorithms are illustrated by computations on the model in Fig. 9.

8.5.5.1 Notations and Definitions

The Bayesian RN-Metanetwork in Fig. 14 has parameters:

+ 1st level:

- This level has only one set of Bayesian networks: 2 Bayesian networks modeling the preferences of 2 users and one Bayesian network for the case of no user.

- The attributes of the Bayesian networks: Time (denoted T), Location_A/B (denoted Li, i=1, 2 corresponding to A and B), CurrentActivity _A/B (denoted CAi), ServiceCategory (denoted Sx, x=1, 2, 3 for its duplications in 3 networks) and has the values {SV1, ...SVj}.

- The prior probabilities: P(T); P(Li); P(CAi), P(Sx).

- The conditional probability P(Si|T, Li, CAi), i=1, 2 and P(S3|T).

+ 2nd level:

- The attributes: Present_A/B (denoted Pri) represents the Presence of user A/B with the values {yes/no}; Pr denotes the set of all Pri.

- The relevance node: Used_BN (denoted BN) holds the probability to have each Bayesian model in the predictive level with the values {BN1, ... BN3}.

- The prior probabilities: P(Pr) denotes the probability distribution of the set Pr and P(Pri) denotes that of each item in the set.

- The relevance probability: P(BN). The conditional probability P(BN|Pr)

8.5.5.2 Probability Propagation

Given the evidence P(Pr), P(BNx) is calculated as:

$$P(BN_x) = P(BN = BN_x \mid \Pr).P(\Pr)$$

Then the probability of the target attribute ServiceCategory can be estimated:

$$P(S) = \sum_{x=1}^{3} P(BN_x).P(S_x)$$

with P(Sx) calculated given the evidence T, Li and CAi

$$P(S_x) = \begin{cases} P(S_x \mid T, L_x, CA_x).P(T).P(L_x).P(CA_x) & x = 1,2 \\ P(S_x \mid T).P(T) & x = 3 \end{cases}$$

In other words, the probability of ServiceCategory preference of each user will be calculated separately, and then combined with the weight coefficients which are the distributed probabilities of the Bayesian networks in Predictive level.

The target ServiceCategory SVj with highest P(S=SVj) will be selected.

The Bayesian RN-Metanetwork provides an easy but efficient method for modeling many kinds of user preferences, from multimedia services such as music, television, radio, web page, public information, etc. to the environment parameter such as light, temperature, etc. All we have to do is identify the features which affect the preference of users, build the preference model for each user (a rather simple task if we consider the current development of Bayesian network research) and finally combine them by a RN-Metanetwork. When the system learns, each preference model is updated separately so that they can be reused in other systems.

8.5.5.3 Adaptation

Even though some approaches address learning issue as the initial of the conditional probabilities from example data sets, the true meaning of learning in a ubiquitous system is online learning. When a ubiquitous system starts working, there is no example data but only the domain knowledge and user-defined rules. For example, the contextual level network can be initialized by the users' initial priorities. However, the task of online learning, or adaptation, is crucial.

Each time the system makes a decision about which service category to be selected, it then wait a time interval t_w for the response of users. There are 2 cases of user responses:

Approval: If there is no response, then the system assumes that the users are pleased with the decision. In this case, the decision together with the evidences will be considered a single sample and be used to update the Bayesian networks conditional probabilities.

Denial: If one user gives a control command to the system to change the selected service category, it means that the user may not satisfy with the decision of the system. This is a serious case, and the system should not make the same wrong prediction again. So the user's selection together with the evidences will be considered N samples (N>>1) and be used to update the Bayesian networks conditional probabilities.

The adaptation algorithm is based on two assumptions:

Assumption 1: *The contribution of one user preference in the combined preference is equal to his contribution in making a decision.*

The assumption means: When the system estimates the preference of many users, it gives each user's preference a weight. In the other hand, whenever a control command is given by a group of user, each user has his contribution in that command. The weight in first case and the contribution in second case are assumed to be equal.

Assumption 2: *Every user has the tendency of selecting the option which has highest probability calculated by his preference model. This option is called the most favorite option.*

The assumption means: Given the evidences, calculate the posterior distribution of a preference using the preference model of one single user (one of 2 Bayesian networks for 2 users A and B in the previous example, for instance), the option with highest probability can be considered that user's most favorite, and is most likely to be selected by the user himself.

Assumption 2 leads to a definition:

Definition: *a decision matches user preference if it matches the most favorite option of that user.*

Based on the above assumptions and definition, the adaptation algorithm for RN-Metanetwork is introduced:

Adaptation Algorithm

Step 1: update the meta-layer network
- In Approval case, there is no need to update the meta-layer network (users satisfy with the current priority)
- In Denial case:
+ Find the most favorite option of each user

$$SV*_i = \arg\max(P(S_i))$$

where SV*i denotes the most favorite service of user i.
+ Count 1 for the value of relevant node related to the user's preference model if the final decision is the same with user's most favorite option. We use the sequential updating introduced in [30]. Do the following adaptation:

If SV*i = SV* then

$$P(BN_i \mid \Pr_e) = \frac{P(BN_i).s.q + 1}{s}$$

And

$$P(BN_j \mid \Pr_e) = \frac{P(BN_j).s.q}{s} \quad \text{with } j \neq i$$

with SV*: the finally selected services, Pre: the set of evidences of the presences of users, q: the fading factor, q (0,1), s: the effective sample size which is calculated by:

$$s = \frac{1}{1-q}$$

Step 2: update the preference model of each user based on the contribution of that user into the decision

For each user i: Calculate wi, the distribution probability of that user's preference model: $w_i = P(BN_i)$. Count wi for the final selected option and update the user's preference model 1 or N times, in approval or denial case respectively.

$$P_i(S_i = SV^* \mid T_e, L_{ei}, CA_{ei}) = \frac{P_i(S_i = SV^* \mid T_e, L_{ei}, CA_{ei}).s.q_i + w_i}{s}$$

And

$$P_i(S_i = SV_j \mid T_e, L_e, CA_e) = \frac{P_i(S_i = SV_j \mid T_e, L_e, CA_e).s.q_i}{s} \text{ with } SV_j \neq SV^*$$

where Te, Lei, CAei denote the evidences of the Time, user location, user current activity of user i, Pi(S | Te, Le, CAe) denotes the distribution of conditional probabilities in preference model of user i qi: the fading factor which is calculated separately for each preference model, to maintain the same experience size.

$$q_i = \frac{s - w_i}{s}$$

8.5.5.4 Probability Propagation and Adaptation for multi-layer Bayesian RN-Metanetwork

In nature, multi-layer Bayesian RN-Metanetwork is a 2-layer Bayesian RN-Metanetwork with the meta-layer is a Bayesian RN-Metanetwork itself.

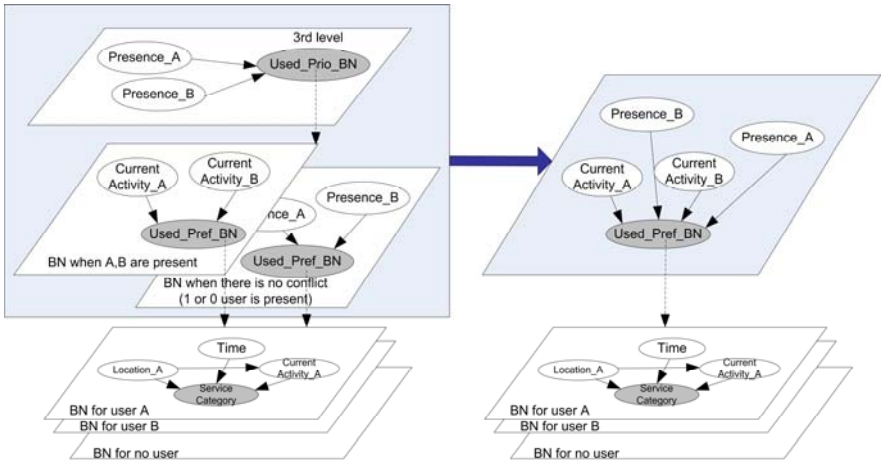


Figure 12: The first 2 levels of a 3-level Bayesian RN-Metanetwork equals with a Bayesian network with same input and output nodes and the intermediate node User_Prio_BN is omitted.

Therefore, the probability propagation algorithm and adaptation algorithm which were described in the previous section can be applied for multi-layer Bayesian RN-Metanetwork in a recursive manner.

The probability propagation process for the Bayesian RN-Metanetwork in Fig. 10:

Step 1: applying the probability propagation algorithm for the first 2 layers to calculate the marginal of User_Pref_BN

Step 2: use the marginal of User_Pref_BN as the distribution for the Bayesian networks in third layer to calculate the marginal for Service_Category.

The adaptation process for the Bayesian RN-Metanetwork in Fig. 10:

Step 1: in case of denial, applying the adaptation algorithm for the first 2 levels of the Bayesian RN-Metanetwork.

Step 2: calculate the marginal of User_Pref_BN and use it to do adaptation for the third level.

In general, the recursive probability propagation and adaptation mechanism for multi-level Bayesian RN-Metanetwork are illustrated in Fig. 13.

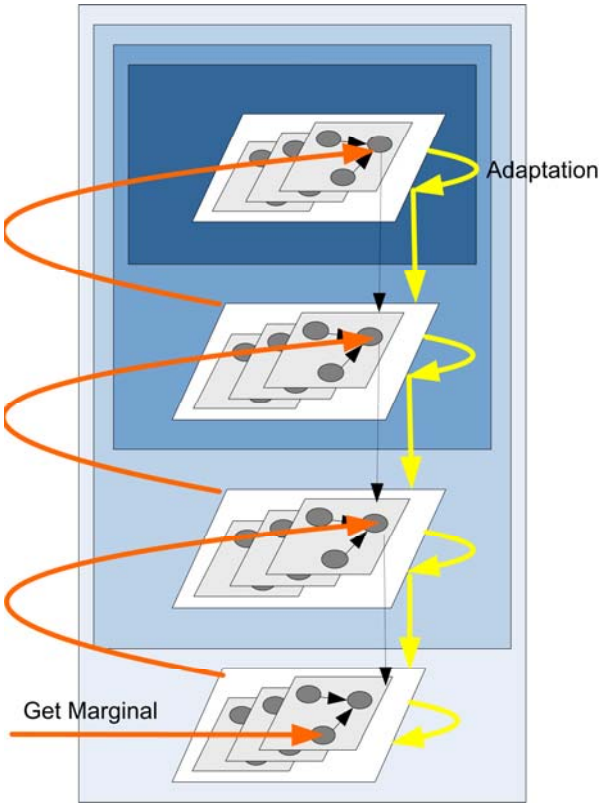


Figure 13: The recursive probability propagation and adaptation mechanism for multi-level Bayesian RN-Metanetwork.

8.5.6 Evaluation

One can argue that the traditional Bayesian network is still faster than the Bayesian RN-Metanetwork, because for Bayesian RN-Metanetwork we have the overhead of the meta-layer propagation, as well as all the preference models and the priority models should be propagated. In fact, the Bayesian RN-Metanetwork is slower just in case the distribution of every Bayesian network in the first layer is not equal to 0, or by other words, when all users are presented in the system.

Let estimate the calculation time for the Multimedia preference example above when using the traditional Bayesian network and Bayesian RN-Metanetwork.

Analysis in [31] shows the runtime for the brute force method of enumeration is $O(q^m)$, where q is the size of the alphabet (in our example: q = number of values for Service Category i for the preference model, or number of users + 1 in the meta network) and m is the number of unknown variables.

We have seen that Pearl's algorithm, for the special case of a polytree, has an efficient runtime of $O(Nq^e)$, where e is the maximum number of parents on a vertex [32]. It can be seen that in that in the case of the turbo-decoding algorithm [33], the runtime is linear in the size of the network, as evidence is propagated a constant number of times.

Here because both the number of unknown variables and the number of maximum parents will increase when the node number increase, we assume that in general the runtime is $O(q^x)$ with x is proportional to the size of the network.
From Fig. 12, the size S of the network in traditional case is proportional to the number of user. So with N = number of user and q = number of values for Service Category i for the preference model, the propagation time is $O(q^{\alpha N})$ with α is proportional to the number of node in a single user preference model.

From Fig. 13, the size S' of each preference network in Bayesian RN-Metanetwork remains the same for every user. S' = S / N. We have the propagation time of each preference network is $O(q^{\alpha})$ and the propagation time of meta network in Fig. 13 is O(1) given that we know the location of all the users.

Then the numbers of preference model should be propagated k has the binomial distribution, because this is the distribution of obtaining exactly k (Presence = Yes) out of N trials.

$$P(k \mid N) = \frac{N!}{(N-k)!k!}.p^k (1-p)^{N-k}$$

Where p is the mean probability of one user is presented in the location.
The expected value of k:

$$E[k] = N.p$$

Then the expected propagation time is: $O(Npq^{\alpha})$

We have $O(Npq^{\alpha}) < O(q^{\alpha N})$. The Bayesian RN-Metanetwork is still more efficient.

In case of situation-dependent priority, the propagation runtime is added with the time for calculating the priority given the evidences about situation.

The runtime for each priority model is $O(q^{\alpha N^*})$ with N* is the mean of the binominal distribution of taking k users from N users. We have N* = N.p. Hence the expected propagation runtime for a priority model is $O(q^{\alpha Np})$.

Then the total propagation runtime is:

$$O(Npq^{\alpha} + q^{\alpha Np})$$

In this case, it is hard to tell whether the Bayesian RN-Metanetwork is faster or slower. However, besides the speed, the advantages of Bayesian RN-Metanetwork come from at least two aspects:

- First, with the division of a large network into small and single-user models, the design of any Bayesian RN-Metanetwork becomes much easier, especially when we have to assign the conditional probabilities for the network.

- Second, we have the separate models for priority and preference. We also have separate models for each user. This dramatically increases the reusability of the models.

8.5.7 Implementation

To illustrate the use of this proposed approach, we will describe in details some models of user preference, which are implemented in Hugin Tool [39]. These are the most common user preferences in smart home systems, such as the preference for light, temperature, curtain, multimedia service category. The models go along with some numerical examples to show how they work.

This chapter will also contain the implementation details for the Bayesian RN-Metanetwork, as well as the user preference learning mechanism in CAMUS.

8.5.7.1 Implementation of User Preference models for Smart-Home Systems

Every system has its own input-output and hence has specific kinds of user preference. One of the famous scenarios is a smart home system where various sensors, controls (actuators) are installed, and many users are involved. In our implementation, we assume that there are 2 users in the house. The preference and priority models which are presented in this chapter are implemented in Hugin, a tool for building Bayesian networks [39].

Categories of User Preference in a Smart Home System

As previously said, user preference is what the user want the system to do in certain situation. In more details, it is the relation between the context of user, which consists of sensor data and other input information, and the control commands which the system give to the actuators inside it. Therefore, to know what kind of user preference we have in a system, we should know all the input information / sensor data as well as command controls a system should have.

The preference learning module in a Smart home system takes as it input the time, the location of user, home environment sensor data, including indoor/outdoor light intensity, audio intensity, indoor/outdoor temperature, current weather, etc. It also receives the current activity of the user from the activity reasoner, which can be implemented using a Bayesian network [8]. Light preference may consider the affection range of each light, while Temperature preference takes into account the duration of user staying in the room

(for example, if the user has just enter the room and the outside is hot, he will need cooler temperature than the user who has stayed in the room for a long time.

The outputs of the preference learning module in a Smart home system are the preferred light intensity level (commands for the light control), temperature level (commands for air conditioner and heater control), multimedia service categories (such as television channel, music genre and audio broadcasting channel), curtain control (close / open), etc. Because the multimedia service has already discussed above, in this section we will only describe the models for curtain control, light and temperature preference.

There is an assumption that there is no body sensor to tell about the user health status or emotions. Otherwise, the preference will depend a lot on user emotion and health.

Curtain Control Preference

In some systems, a curtain can have only 2 statuses CLOSE and OPEN. In other system, the curtain control can open the curtain to some levels. Assume that we have 3 levels for curtain control: CLOSE, OPEN and HALF_OPEN.

Normally, user wants to open or close the curtain depending on the outside weather, light intensity, sunlight direction and the activity which is happening in the room. For the weather, we can have some rain sensors to detect rain, and sunlight sensor to detect whether the sunlight enters the room or not.

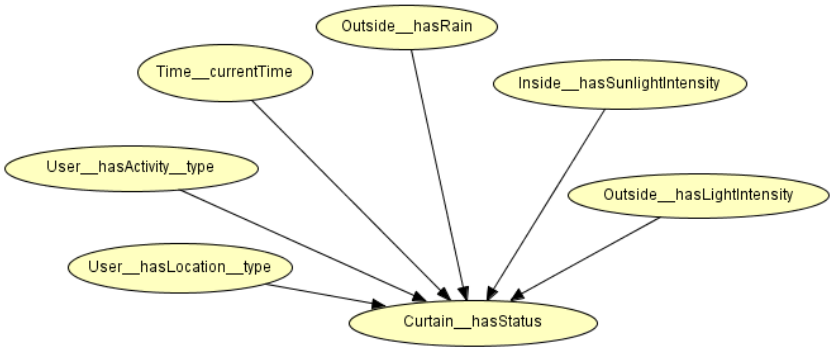


Figure 14: The curtain control preference model of one user

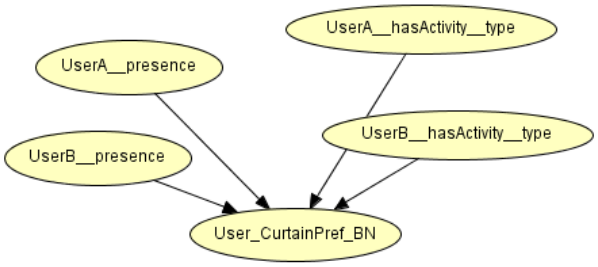


Figure 15: The priority model for curtain control preference (for 2 users) using traditional Bayesian network.

The priority of curtain control is related to the presence and the current activity of the user.

Initial the priority model:

To initialize the CPT, we can learn the conditional probability from the history data, or enter the probability manually, or use expression. This is an example of Hugin expression for the CPT of User_CurtainPref_BN node:

```
if (and (UserA__presence == "NO", UserB__presence == "NO"),
Distribution (0, 0, 1), Distribution (0.5, 0.5, 0))
```

The Distribution function in this expression is the distribution of the states of User_CurtainPref_BN node, which are BN1, BN2 and BN0. These states are the preference models of user A, user B and the model in case no user is presented.

This expression gives the equal priority for both users without concerning the current activity of each user. If we allow the user who is sleeping to have higher priority, the expression would be like this:

```
if (and (UserA__presence == "NO", UserB__presence == "NO"),
Distribution (0, 0, 1),
if (and (UserB__hasActivity__type == "SLEEPING",
 not (UserA__hasActivity__type == "SLEEPING")),
Distribution (0.2, 0.8, 0),
if (and (not (UserB__hasActivity__type == "SLEEPING"),
UserA__hasActivity__type == "SLEEPING"),
Distribution (0.8, 0.2, 0),
Distribution (0.5, 0.5, 0))))
```
[1]

Using the function “Transfer to table” of Hugin [39], we will have a CP table, a part of which is showed in the following figure.

UserB__h...	WORKING								SLEEPING							
UserA__h...	WORKING				SLEEPING				MOVING				WORKING			
UserA__pr...	YES		NO		YES		NO		YES		NO		YES		NO	
UserB__pr...	YES	NO	YES	NO	YES	NO	YES	NO	YES	NO	YES	NO	YES	NO	YES	NO
BN1	0,5	0,5	0,5	0	0,8	0,8	0,8	0	0,5	0,5	0,5	0	0,2	0,2	0,2	0
BN2	0,5	0,5	0,5	0	0,2	0,2	0,2	0	0,5	0,5	0,5	0	0,8	0,8	0,8	0
BN0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1

Figure 16: CPT of the curtain control priority model

Light Control Preference

Most lights have 2 common states: ON and OFF. Assume that there are 4 levels of light intensity in the room: BRIGHT, NORMAL, DIM and DARK. The intensity levels which each light can produce range among BRIGHT, NORMAL and DIM.

[1] The expression is in the format of Hugin expression, building by the Hugin tool [39].

The light control preference of one user will be related to the current activity of all the users involved, current light intensity in the room, outside light intensity and the curtain status. All the lights which have the same affective range with the user location will be considered in the model.

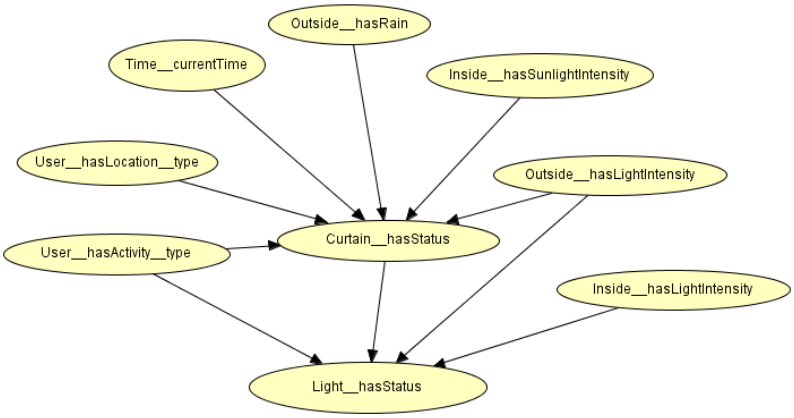


Figure 17: The Bayesian network models for the light control preference of one user

The priority of the light intensity control also depends on the current activity of user. For example if the user is sleeping, light should not be bright.

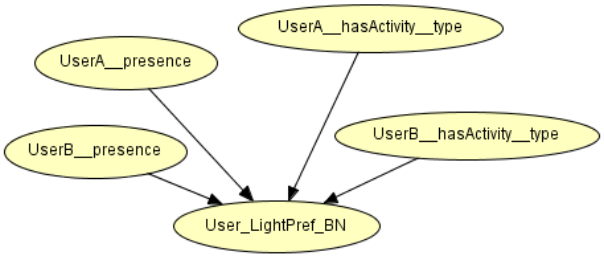


Figure 18: The priority model for curtain control preference (for 2 users) using traditional Bayesian network.

The initial of priority model is similar to the previous session.

Temperature Preference

Every air conditioner has its own types of control. However, the control can be mapped into some certain level of temperature. Assume that there are 3 levels: High, Moderate and Low.

The temperature preference of user depends on inside and outside temperature, user activity and the occupation duration. The temperature preference priority depends on the occupation duration. A user who has just been walking outside needs the help of the air conditioner or the heater more than a user who has stayed in the room for a long time.

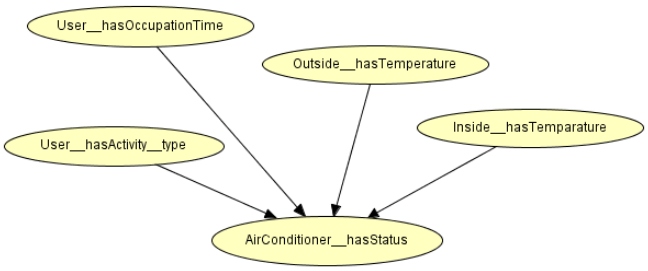


Figure 19: The Bayesian network models for the air conditioner control preference of one user

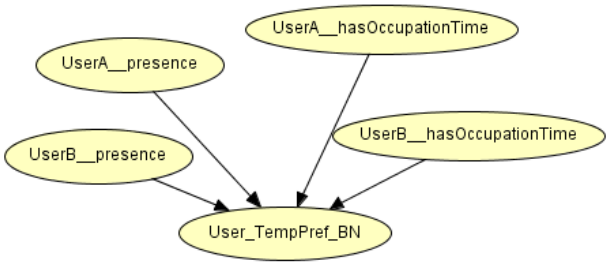


Figure 20: The priority model for air conditioner control preference (for 2 users) using traditional Bayesian network.

8.5.7.2 Adding more user preference models into a Bayesian RN-Metanetwork

When a new user joins the system, the preference model should be updated. The process for integrating a user into a multi-user preference model has 3 steps:

Step 1 (optional): create the preference model for that user based on the common preference model for a single user.

From the common preference model, we make a copy and then rename the nodes related to user to the name of that user.

For example, the air conditioner control preference model for UserA will be:

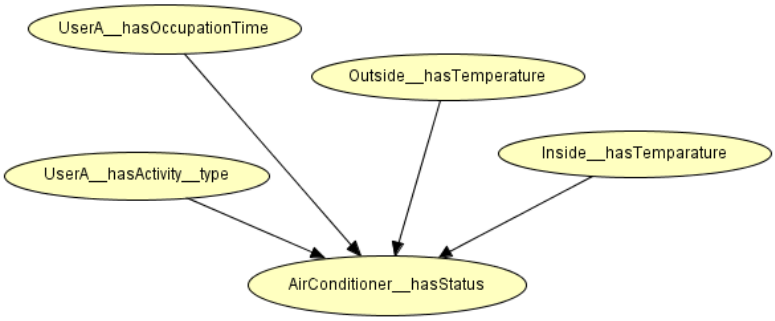


Figure 21: Air conditioner control preference model of UserA

This step can be omitted if the user preference model has already built before.

Step 2: add the new preference model into the set of preference model in the predictive level (first level) of the Bayesian RN-Metanetwork.

Step 3: update the priority model:

- Add the nodes related to the new user into the priority model

- Add one more state which is equivalent to the new user preference model

- Update the CPT:

+ Where new user presence has value NO, the conditional probability can remain the same, and 0 is filled for the probability of the new state.

$$P(BNN|X,Y,PrN=NO) = 0$$
$$P(BNi|X,Y,PrN=NO) = Po(BNi|X)$$

with N denotes the current number of users,
X denotes the set of nodes which related to old users
Y denotes the set of nodes which are related to new user except the presence,
PrN denotes the Presence of user number N,
P: new conditional probability
Po: old conditional probability

+ Where the new user presence has value YES, the conditional probability is recalculated. Normally the calculation is based on the priority mechanism of the system. If the system is first-equal-priority, i.e. all at the beginning all the users have same priority, the calculation will be:

New state:

$$P(BN_N|X,Y,Pr_N=YES) = \frac{1}{N}$$

Old state: the old priority order among the old users is maintained by:

$$P(BN_i|X,Y,Pr_N=YES) = P(BN_i|X)*\frac{N-1}{N}$$

8.5.7.3 Bayesian RN-Metanetwork Reasoner Implementation

The Bayesian RN-Metanetwork is implemented as a reasoner inside CAMUS.

The programming language is Java. The library of Hugin [39] is utilized as a core for Bayesian network computations.

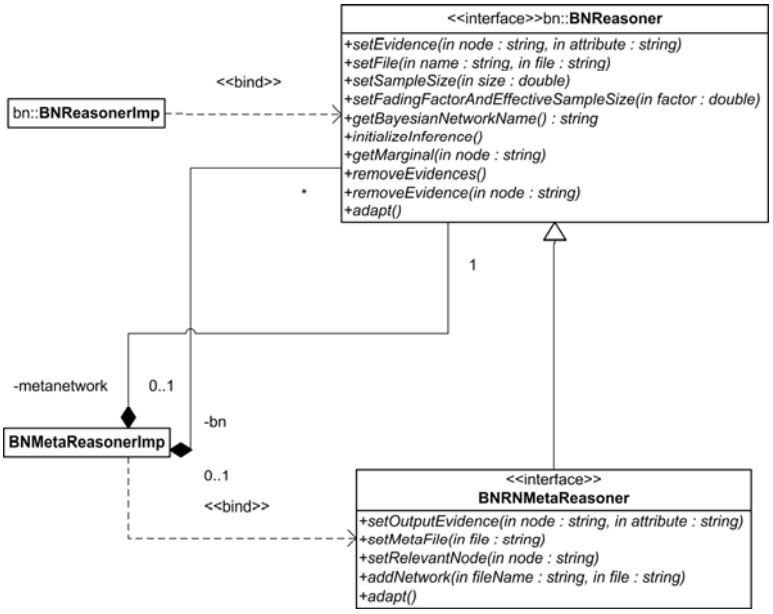


Figure 22: Class diagram of Bayesian RN-Metanetwork

By wrapping the Bayesian RN-Metanetwork in an interface which inherits the BNReasoner interface, the Bayesian RN-Metanetwork reasoners as well as the traditional Bayesian reasoner can be accessed in a unified manner. Because a Bayesian RN-Metanetwork includes the BNReasoners as its metanetwork and prediction-level networks, each of the sub-network can be a Bayesian RN-Metanetwork itself, or in other words, the Bayesian RN-Metanetwork can be multiplied into many layers.

8.5.7.4 User Preference learning implementation in CAMUS

We have a ReasonerManager service to take care of creating the reasoners as well as managing those reasoners. A ContextDataManager manages the context repository, receives the queries and registrations for context data. Inside the ContextDataManager, the rule-based reasoning is implemented to infer the high-level context data from low-level context data. When new context data comes from sensors, the ContextDataManager service does the inference for high-level context data, then matches the registrations and notifies the handler services if the registrations are matched.

A PreferenceContextAggregator service works as the preference learning module for the middleware. It reads the model files, calls the ReasonerManager service to create the Bayesian RN-Metanetwork reasoner, and registers to the ContextDataManager service for the needed context data. When it is notified about the new context data, it will give the new evidence for the Bayesian RN-Metanetwork reasoner by setEvidence method, then asks the reasoner to do inference and finally gets back the marginal for the nodes related to the preferences.

Any application can send request to PreferenceContextAggregator service to get the preference of user. After making the decision and receive the feed back from user, the

application will tell the PreferenceContextAggregator service the result of feedback. Based on user feedback (approval or denial), PreferenceContextAggregator service will set the evidence and call the Bayesian RN-Metanetwork reasoner to do adaptation.

Fig. 23 shows the sequence diagram of all process.

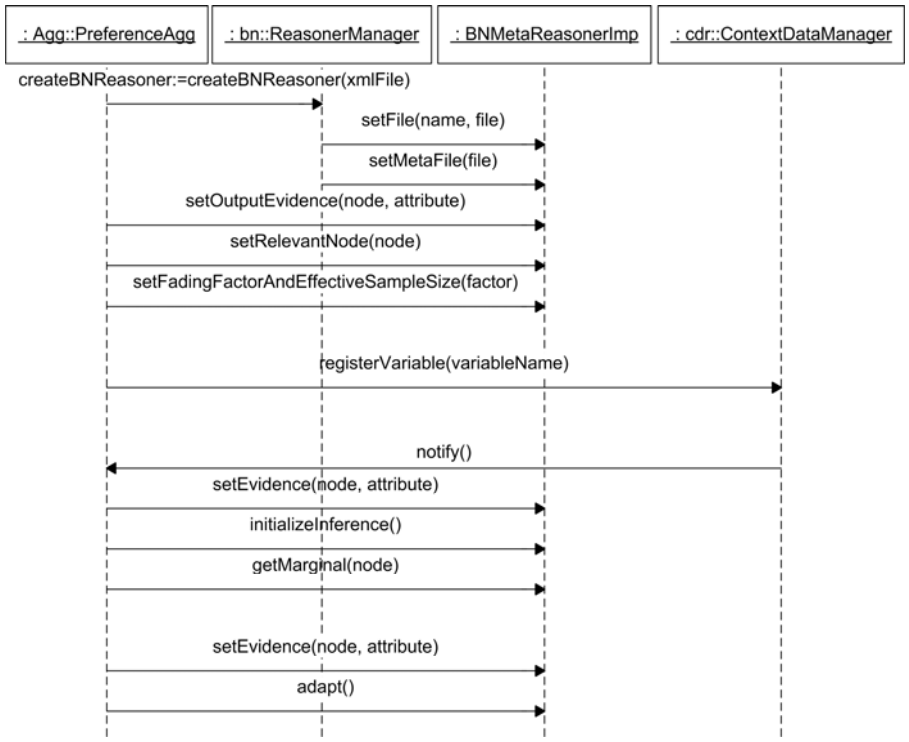


Figure 23: The sequence diagram of user preference learning process in CAMUS

8.5.8 Conclusion

Context-aware computing poses interesting issues for information system researches. Among those issues, learning user preference in order to adapt the system automatically to the need of user is a crucial task. Many challenges have risen in this are due to the uncertain, heterogeneous, distributing characteristic of a context-aware system. Especially when there are many users involve in an intelligent environment, the system has to cope with conflict resolution and distinguishing among the user preferences. A solution for learning user preference in a multi-user context-aware environment which can efficiently resolve the above mentioned problems is the main contribution of our work.

We have first presented the Bayesian RN-Metanetwork which can be used to model the user preference as well as user priority for many users, while still maintain separate preference model for each user. The propagation algorithm showed how to calculate the composite preference of all the users in the system and make decision of service to provide. To actively and continuously adapt the models to the newest preferences and priorities of the users, the adaptation algorithm for Bayesian RN-Metanetwork was

described. To illustrate the use of Bayesian RN-Metanetwork, some common models for various kinds of user preference in a smart home system were presented.

However, this is just the first step to make the system intelligent. There are still a lot of challenges such as user behavior routine learning, in which the prediction of the future actions of user is the most important task.

8.6 Location Estimation

8.6.1 Location Estimation in CAMUS

Emerging mobile computing application and ubiquitous computing environments are required to provide location sensitive behavior to end users. Consequently location systems are on the verge of becoming a part of daily lives of users by enabling a variety of smart environments like shopping malls, campuses and airports. Advances in sensing technologies have resulted in several novel positioning techniques to support many potential location based applications. Location aware computing space is defined mainly by these three complementary technologies as shown in Figure 1.

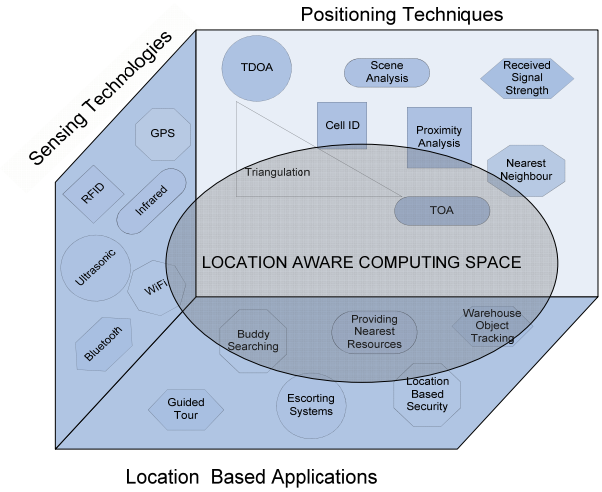


Figure 1: Location Aware Computing Space

This technical report illustrates on Location Estimation research and development in the framework of CAMUS. Main focus of this research is localization in wireless sensor networks particularly indoor location estimation using WiFi networks. Major contributions are providing an integrated middleware infrastructure for developing location systems, specialized machine learning methods, software tools, algorithms and real life implementations of wifi location systems in large indoor environments.

In building localization has been subject to costly infrastructure and special hardware devices mounted on the objects of interest. However, signal strength based location estimation is an attractive choice because of its economic viability and pervasive availability of IEEE 802.11 wireless networks, also called WiFi networks. Received Signal Strength (RSS) based location awareness applications include, but are not limited to, a wide range of services to the end user like automatic call forwarding to user's location, robotic global localization, exploration and navigation tasks, Finder, Guiding and Escorting systems, first hop communication partners, liaison applications, location based advertisement and positioning of entities in large warehouses.

8.6.2 WiFi Location Systems Overview

Location systems form a fundamental component of envisioned ubiquitous computing applications. Several location based services are poised to enrich the way people interact with computers such as activity recognition, personnel management, asset tracking. Global coverage of GPS is considered as most potent example of a ubiquitous location system but it faces no availability of satellite signals in densely populated urban areas in general and especially in the indoor environments. Recently Wireless LAN (WLAN) based location systems have gained a significant attention from research community as well industry; mainly because pervasive availability of WLAN in urban areas and proliferation of wireless network enabled commodity handheld devices.

These WLAN based location systems can be coarsely characterized based upon two aspects as summarized in Table 1; i) The resolution of location estimate and ii) The *prior* knowledge which the system requires as prerequisite.

	Prior Information Required for Learning	
	Position of Tx + Environmental Model	Position of Rx
Coarse Grained 25~50 m Outdoor	Place Lab Intel 2002 LaMarca 2004	
Fine Grained ~3 m Indoor	Rutgers univ. 2006	RADAR Microsoft 2002

Table 1: Taxonomy of WiFi Location Systems

Location Resolution Based Division

Coarse grained location systems, such as Intel's Place Lab, provide 20 to 50 meter accuracy and are more suitable for outdoor scenarios, whereas fine grained location systems, such as Ekahau, are befitting for indoor environments and achieve up to 3 meter accuracy.

Prior Knowledge Based Division

On the other hand, the division based on *prior* knowledge is present in both coarse grained and fine grained systems. Some systems require a detailed radio map of target area which provides the basis for developing a mapping function between physical space and signal space. Since creation of radio map is human intensive task, another class of systems avoids manual creation of radio map by using sophisticated radio wave propagation models. Nevertheless these models require detailed information about the position of WLAN access points, building structure, materials and obstacles; which, needless to mention, is often not easily accessible.

Middleware Infrastructure for WiFi Location Systems

Although ad-hoc location aware applications can be developed without middleware support but this approach results in non reusable effort and monolithic systems. Developing reusable abstractions and services as middleware masks a broad range of issues from location aware system developers. Most of previous research is concentrated on seamless integration of multiple location technologies through a unified programmable interface and provide support for several applications that require tracking, location base publish subscribe and location sensitive customization of end user applications to name a few. In this paper we present a middleware infrastructure for developing both signal strength based location systems using neural networks and support for location aware applications through reflective component framework. Locationware provides interfacing with other location technologies which separates development of location aware application from underlying location sensing mechanisms.

This section presents a conceptual overview of Locationware Middleware and its infrastructural components for end to end support of location aware computing with respect to a coherent development model. This model consists of a well defined stairway development life cycle which guides the phase-wise development of location systems in two logically successive and complementary stages. In the first stage a location system is developed and made available to the subsystems of second stage. The second stage concerns with design, development and execution of end user location-aware applications. Figure 2 shows the schematic of Locationware development life cycle. Each stage of development lifecycle is further divided into a continuum of phases such that the outputs of each phase become inputs to the next phase. During each phase our middleware provides software artifacts, services and tools to design and implement location systems and, subsequently, location-aware applications.

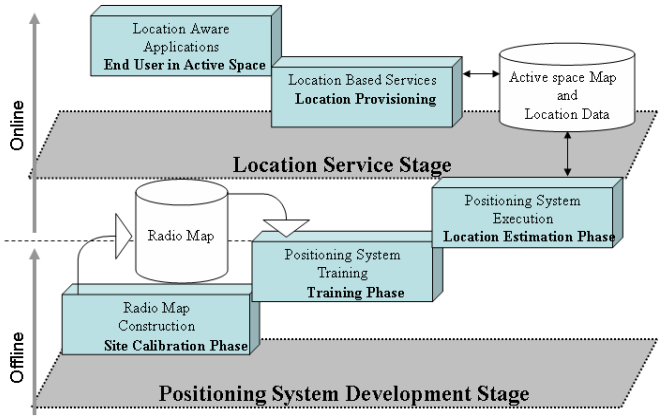


Figure 2: Conceptual Schematic of Development Life Cycle

Figure 3 shows Locationware middleware infrastructure that allows developers to manage and realize development process of location aware system phase by phase. Each phase shown in the conceptual schematic of location system development life cycle, Figure 2, has corresponding subsystem implementation in the Locationware infrastructure shown in Figure 3. We employed principles and practices of component based software

engineering in order to facilitate reusability and upgradeability. For the sake of clarity and coherence with stairway lifecycle, we shall divide the discussion of middleware features into two categories.

i.) Location system development subsystems are briefly described here while detailed discussion is presented in section 4. The rapid site calibration framework realizes site calibration phase by providing a distributed sampling system for parallel and rapid site calibration by multiple devices. Site calibration phase delivers the 'Radio Map' database that is employed to train pattern recognition machines in the next phase. Training phase constructs classifier model, called positioning engine, of target site which learns the mapping of signal strength patterns with corresponding locations. Once the positioning engine is trained, we deploy it into the middleware as software component for runtime location estimation. In order to ship the positioning engine from offline and lab development to online and real life execution; we offer a runtime environment and server side APIs hosting location estimation engine.
ii.) Location-aware applications development concerns with distributed location aware system which provides server side mobile client proxy hosting services, seamless location provisioning to LBS (location based services) and variety of communication schemes to collaborate with the end user location aware applications. Locationware provides a reflective component framework and middleware services to realize location aware application development and on the fly adaptation. This paper focuses on only Location System Development stage, however the details of location-aware application development can be found in [17] and [18].

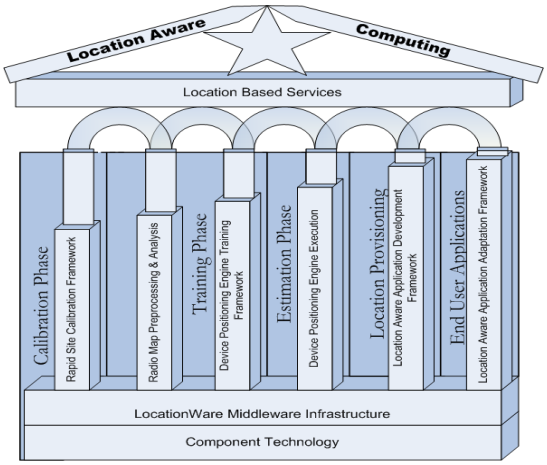


Figure 3: Locationware Middleware Infrastructure

8.6.3 Radio Map Knowledge Engineering Toolkit

In the course of location aware computing research, we developed a complexity analysis toolkit for preprocessing the Radio Map. This toolkit provides rich set of statistical analysis functions such as Linear Discriminant Analysis, Most Effective Feature Selection, Scatter Analysis, Identification of Overlapping classes and there graphical representation using parallel coordinates and Feature Space Transformation functions for

different classification techniques such as Learning Vector Quantization and Multilayer Perceptron. Figure 4 shows a snapshot of the GUI of Radio Map Knowledge Engineering Toolkit.

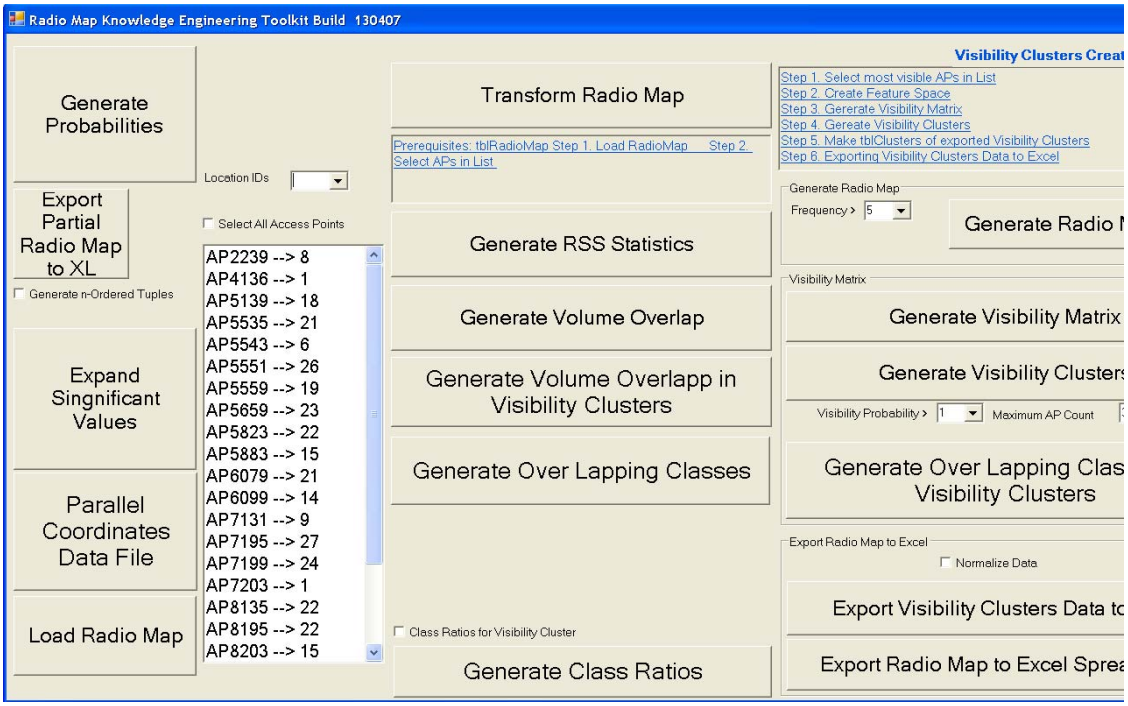


Figure 4: Radio Map Knowledge Engineering Toolkit

8.6.4 Parallel and Distributed Site Calibration System

The relationship between site calibration effort, system complexity and target granularity of localization system is very straightforward. Since radio map plays the role of reference map for localization techniques, the accuracy of location estimation is directly linked with the correctness of radio map. As the measurements of RSS become less reliable, the complexity and error of the position algorithm increases. Therefore rapid and accurate calibration of target site is established as major issue in adoption and deployment of Location Aware systems. We present a distributed and parallel computing approach to expedite development of comprehensive radio maps for Location Awareness. A novel method of RSS data collection is presented. Results show noticeable improvement in calibration speed and reduction in radio map database size. Location-ware essentially provides an API and framework for implementing distributed system in a highly dynamic mobile computing environment. Moreover same API can be used for testing and evaluation of a positioning system. Since Wireless LAN based positioning techniques are strongly relative to a particular site and no general system can be developed for all wireless sites. Testing and evaluation task faces with the same problems as site calibration. So the same distributed and parallel computing approach can be employed for gauging the performance (accuracy and precision) of a positioning system. In section 4, we elaborate the API and reference implementation of Location-ware. Besides, our system allows real time visualization of RSS of all available wireless base stations, giving

more insight into the nature of wave propagation at different locations. Thus, providing an efficient way to analyze the properties of WLAN and achieve optimal WLAN coverage, enhancing the positioning system performance. Besides, we developed a novel method for capturing and storage of RSS data. Since the radio map size and complexity increases exponentially as the area of interest becomes larger. We integrate histogram based data collection method in calibration system that reduces radio map size without compromising on the quality of radio map.

8.6.5 System Architecture

Location-ware architecture follows distributed component approach to realize the concept of distributed and simultaneous site calibration. Whole system is divided in three sub systems. I) Calibration Agent: deployed on handheld devices, ii) Calibration Server: deployed on stationary workstation iii) Radio map: provides target database wrapper API, and radio map manipulation components.

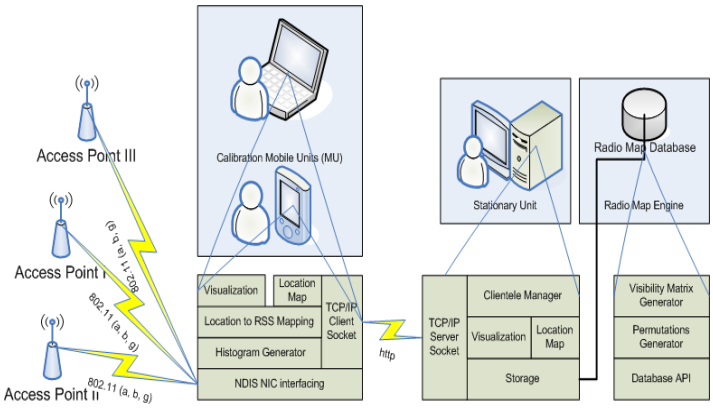


Fig. 5. System Architecture Building Blocks

Calibration Agents

We refer to the subsystem that resides on hand held device as Calibration Agent. Although these components can be deployed on other small handheld devices, for actual calibration task we used TOSHIBA M-30 notebook with built in Intel PRO/Wireless 2200 BG Network card and HP iPAQ Pocket PC h4150. Calibration Agent sub system API are shown in Figure 6. Calibration Agent allows users to perform different tasks related to calibration. Fig. 7 shows use cases that specify the user – system interaction scenarios. As we discussed in section 2, taking few samples at location is not sufficient enough for building a reliable location estimation technique for real world scenario. Insufficiently calibrated point might lead to misleading conclusions about the location at the time of positioning.

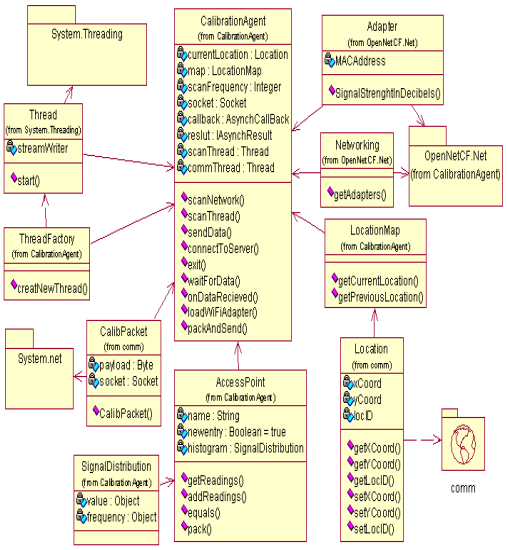


Fig. 6. Calibration Agent UML class diagram

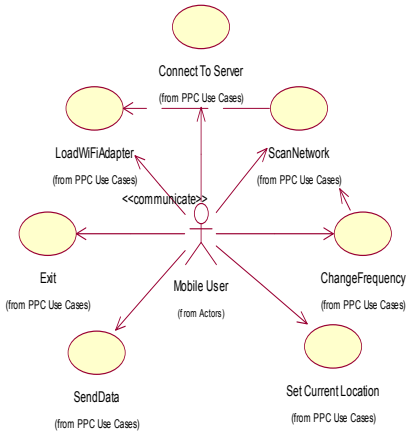


Fig. 7. Calibration Agent Use Cases

'Scan Network' use case allows user mode applications to start the process of scanning network properties. It sends a probe packet to nearby APs and captures required information from the response packets. Essential capability of a site calibration system is to capture properties of wireless network e.g bssid (Basic Service Set Identifier), rssi (Received Signal Strength Indicator). Commercially available and open source like netstumbler[13] meet this requirement but these systems are more focused on network traffic analysis, intrusion detection, layout management and are not meant for collecting data for location awareness. While site calibration system requires a subset of there capabilities, still these systems fall short of providing capture, storage and analysis of received signal strength data for location awareness. This capability can be implemented as software driver at the top of Wireless Network Interface Card (NIC). This layer hides vendor specific implementation of hardware and allows user mode applications to query NIC in a standard way with uniform accessibility and representation of required information. IEEE 802.11 (a, b, g) specifies that signal strength measurement must be reported by the network interface card (NIC) as part of standard compliance. [4]. The RSSI is measured in dBm, normal values for the RSSI value are between -10 and -100. [18] Capturing wireless signal from available Access Points remains the fundamental task in building site calibration systems. A standard library for signal capture should be employed that can hide different hardware vendor specific details of signal capture. MAC driver then interprets the response frame and takes note of the corresponding signal strengths. Figure 8 shows the list of Wireless Access Points in range of handheld device after a probe. For laptop Calibration Agent Subsystem, we used rawehter API [23] for Windows XP for capturing the wireless packets.

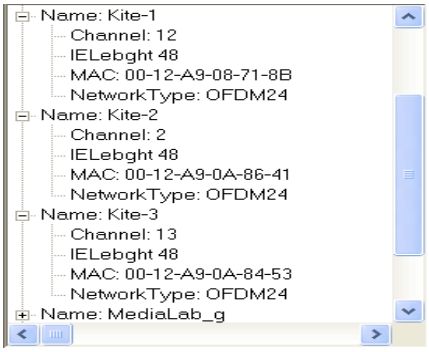


Fig. 8. List of Wireless Access Points

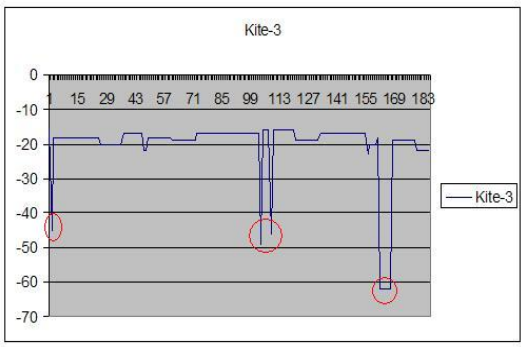
Another unique feature of Calibration Agent network scanning operation is aggregation of all observations into a histogram. Listing 1 shows the histogram algorithm which constructs an rssi data aggregation in Calibration Agent memory.

Listing 1: Histogram making Algorithm

```
1 Create a two dimensional array 'Histogram[m,n]'
2 Create a flag 'newEntry'
3 Create a node 'HistogramSize'
  INPUT: rssi value as integer
  OUTPUT: histogram of rssi values
  For Each count In 'Histogram' Do
      If rssi is same as 'Histogram[Count,0]' Then
          Set 'newEntry' = false
4         Increment Histogram[Count,1] ++
          Return Histogram
      Else
          Set 'newEntry' = true
  If newEntry is true Then
5   Set Histogram[HistogramSize,0] = rssi
    Increment Histogram[HistogramSize,0] ++
```

Algorithm 1: Histogram Algorithm

'Change Scan Frequency' use case allows user to change scanning frequency through GUI. This provision allows signal capture at dynamically configurable intervals. By increasing the sampling interval, one can avoid temporal spikes in signal strength due to environmental factors. Fig 9 shows time series graph of signal strengths at one location from Kite 3 access point. Red circles show the temporary drop in signal strength that can be filtered out using lazy sampling.



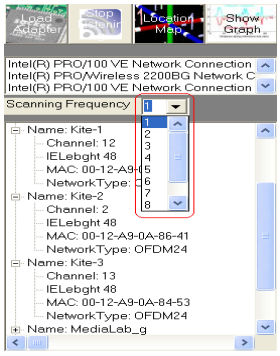


Fig. 9. Temporal spikes in signal strengths **Fig. 10.** Controlling WiFi signal scanning rate

Fig. 10 shows GUI for controlling sampling frequency dynamically. Our scanner component exposes this capability through standard programming interface. Client mode applications can choose their preferred rate of network scanning. This capability is implemented in our system as GUI control.

The Location Map provides a reference map of target area that needs to be calibrated. User specifies his location and Calibration Agent binds the respective location with received signal properties. Calibration Agent provides active location map of the site in order to allow the user to pinpoint the current location of observation.

After calibrating one location, “Connect to Server” use case establishes connection between Calibration Agent and Calibration Server. TCP/IP sockets are used for communication. Network communication component on handheld device registers itself with the Calibration Server and get and MU-ID response from server. Later this MU-ID is used to report WiFi signal information, acquired from all Access Points in range, back to the Calibration Server. At the SU side, network communication component registers every MU roaming in site and manages all the incoming information using client management system using multiple independent threads of communication. Once connection is established, “Send Data” use case allows user to send calibration data of that location to server and let user know about the received acknowledgement from server. Calibration Agent uses Wireless LAN to establish connection and send/receive data to/from Calibration Server. On the other hand Calibration Server is connecter with local Ethernet network. Above this layer both subsystems use TCP/IP protocol to address each other. Sockets layer implements asynchronous mechanism for communication. Asynchronous communication allows distributed collaborations of multiple mobile devices with Calibration Server run smoothly. Figure 11 shows layered view of infrastructure that allows communication between Calibration Agent and Calibration Server.

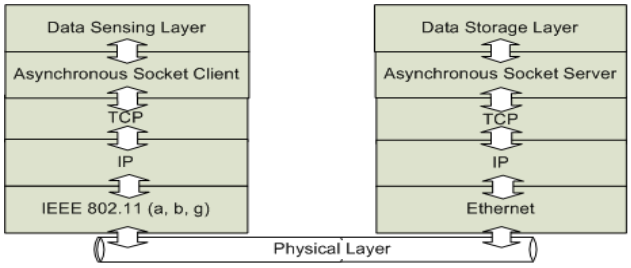


Fig. 11. Layered Model for Communication between Mobile Units and Calibration Server

Calibration Server

On the other hand, stationary unit performs the important task of managing simultaneous incoming information of every MU. Calibration Server has three main components: i) Communication Component ii) Clientele Manager iii) Real Time Charting Component. Communication component is responsible for receiving rss data from multiple roaming devices and send acknowledgement packets to individual device. Calibration server classes and their interrelationships are shown in figure 12 class diagram. CommServer class encapsulates wireless communication details, storage mechanisms, calibration data parsing, parallel client processes and exposes standard functions to the developers for making web browser or other GUI calibration server applications. XL class provides interfacing with excel sheets to store calibration data while Calibration Data class represents a histogram structure for in memory storage of rss data of each access point.

The Clientele Manager component is responsible for registering remote calibration devices in clientele registry and managing communication with individual devices in separate processes. Calibration Server sends an acknowledgement to individual devices containing the receipt of total number of records saved in database on behalf of that particular device. Clientele Manager Component manages this operation and sends respective information to each mobile device. Besides, clientele manager allows broadcasting some message to all mobile devices on behalf of Calibration Server.

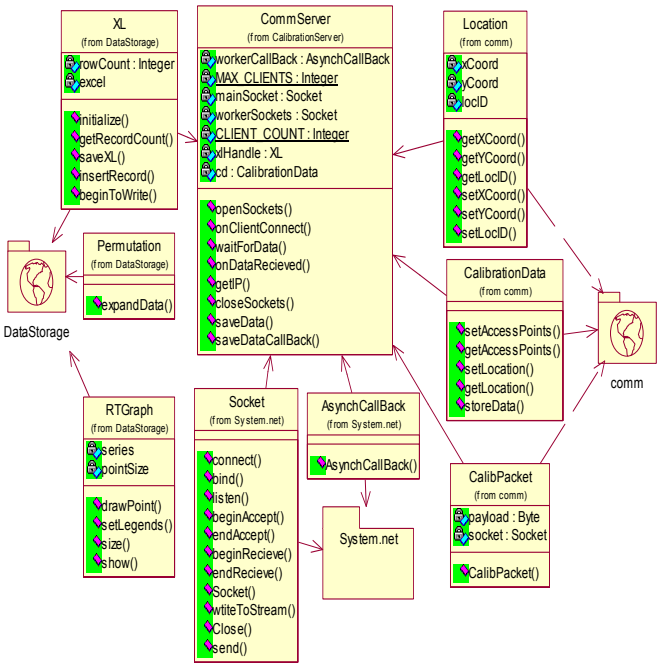


Fig 12. Calibration Server Class diagram

Radio Map Based Location System

We conducted experiments in 1240 square meter area of Engineering Building shown in Fig 13 with target locations. This area is covered by 8 wireless lan access points. For sensor data collection, we employed Hp iPAQ pocket PC devices equipped with integrated Intel wireless network interface card. IEEE 802.11 (a, b, g) standard specifies that signal strength measurement must be reported by the network interface card (NIC) as part of standard compliance. The RSS is measured in dBm and normal values for the RSSI value are between -10 and -100. We collected 150 samples of RSS from all available Access Points at each location in calibration phase for training the classifier. Same size of testing data was collected for testing the location estimation performance. In order to effectively capture noisy characteristics of radio channel due to several environmental factors, both sets of data were collected in 5 days at different times of each day.

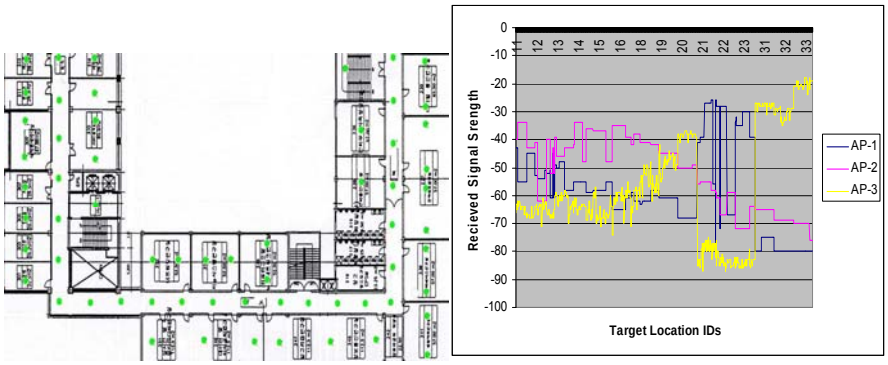


Fig 13: Location Map, Target Locations showing RSS patterns of three

Fig 14: Partial Radio Map access points

Noisy characteristics of RSS values can be seen in Figure 14 which shows signal strength of three access points recorded at subset of target locations. Location IDs are listed on x-axis and RSS values on y-axis. Moreover, Device at two different locations can sometimes report same RSS readings, and can report very different readings while at the same location. Target classifier takes received signal strengths of visible access points as input and generate location as output. This mapping is learned through training phase using training Radio Map. Training phase is responsible for data preprocessing, classifier training, post processing and error analysis and tuning classifier for optimal results. Figure 15 shows system components that are involved in training phase.

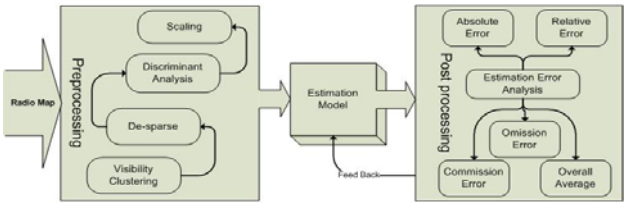


Fig 15: Model Training Phase

During preprocessing step, we apply a clustering technique that extracts prior probabilities of visibility of access points on set of locations from Radio Map. Resulting visibility clusters are used to de-sparse feature space. RSS vectors are then normalized in order to make all values fall in -1 to 1 range. Results show that range normalization improves estimation accuracy of SVM and MLP classifiers. We also applied smoothing filters to remove outliers from RSS patterns. Real RSS values exhibit temporal spikes due to environmental factors. After training phase live data from the environment need to be tested with trained classifiers. In estimation phase RSS captured on mobile device is presented to the input layer of neural network. After the number of accessible AP is determined, different preprocessing components are implemented to filter, scale and normalize data. Figure 16 shows all the components involved in execution phase.

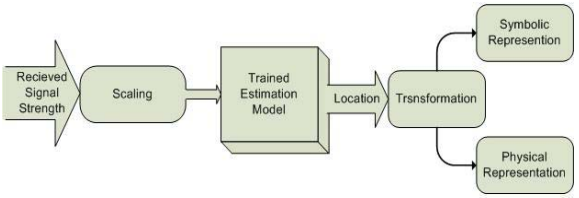


Figure 16: Location Estimation Phase

Offline Learning Methods

Support Vector Machines

Seminal work on support vector machines (SVM) has been rigorously characterized by Vapnik. SVM belongs to kernel based machine learning techniques. SVM have shown superior results in various applications of face detection, object recognition, handwritten character recognition, speck recognition, time series prediction and biometric identification system [21]. The basic idea in support vector machines is to construct a special hyperplane between classes that separates them with largest or optimal margin. In simplest two class problem having input vectors $x_i(i=1,...,n)$ in a N-dimensional input space R^n with corresponding class labels $y_i \in \{-1,1\}$, a SVM classifier is based on class of hyperplanes, defined in (1), where w is the weight vector, x is the training vector and b is bias.

$$(w \cdot x) + b = 0 \tag{1}$$

The decision function takes form as described in (2).

$$f(x) = \text{sgn}((w \cdot x) + b) \tag{2}$$

The geometric measure of optimal margin becomes $1/2(w^T w)$. The optimal hyperplane can be found by minimizing $1/2(w^T w)$ subject to $y_i(w \cdot x_i + b) \geq 1 \forall_i$. Lagrange multipliers are used to solve this optimization problem. That formulates optimal hyper plane finding problem as maximizing following.

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j) \tag{3}$$

Subject to $\sum_{i=1}^{l} y_i \alpha_i = 0, where \alpha_i \geq 0, i = 1,...,l$. Generalized form of SVM solves multi class problems with non-linear separating hyperplanes. It is realized by mapping the input space x_i into a higher dimensional space also referred to as augmented space. Different kernel functions, shown in table 2, are used to achieve this mapping: $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$.

Table 2: Different Kernel Functions

Kernel	Formulation
Linear	uv
Polynomial	$(\gamma uv \oplus c)^{\deg ree}$
Radial Basis Function	$e^{(\gamma \lvert uv \rvert^2)}$
Sigmoid	$\tanh(\gamma uv \oplus c)$

Depending on kernel choice the input space can be transformed into a feature space with linearly separable classes which were non-separable in original space. Even if classes are not completely separable in new feature space, SVM still can construct optimal margin separating hyperplanes by allowing error penalty variables which relaxes the hard margin condition. Tradeoff among different training errors is regularized by $C\sum_{i=1}^{n}\xi_i$ where C is regularization constant. By substituting constant C and respective kernel function, problem equation (3) becomes as maximization of following.

$$W(\alpha) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j K(x_i \cdot x_j) \qquad (4)$$

$$\text{Subject to} \quad \sum_{i=1}^{l} y_i \alpha_i = 0$$

$$0 \le \alpha_i \le C, i = 1,...,l$$

We employed support vector machines LIBSVM allows programmatic customization of two support vector classification techniques (i. C-SVC ii. nu-SVC) as well as configuration of four kernel functions. It also supports multi class problems by employing "one-against-one" algorithm. In this method $k(k-1)/2$ classifiers are actually generated for k classes, where each training vector is compared against two different classes and the error (between the separating hyper plane margins) is minimized. The classification of the testing data is accomplished by a voting strategy, where the winner of each binary comparison increments a counter. The class with the highest counter value after all classes have been compared is selected. RSS based Location system development life cycle can be defined in three distinct phases;

Online Learning Methods

In this direction we developed a novel location system based on self-scalable Fuzzy ArtMap. Fuzzy ArtMap is more general ArtMap (also called Predictive Art) network which can handle analog input patterns and performs online incremental supervised learning of pattern-class pairs presented in arbitrary order. Compared to previous work our location system offers several desirable features which cannot be realized using previous methods. Rapid Development of RSS based location systems is major issue that keeps this technology from becoming widely deployed. Unlike previous systems, this approach requires no online training (or lab time) to train classifier.

Fuzzy Art and Fuzzy ArtMap

We adapt a simplified version of Fuzzy ArtMap, presented in, which employs only one Fuzzy Art network instead of two with same learning and recall performance as original Fuzzy ArtMap. Simplified architecture is optimized for hardware implementation of Fuzzy ArtMap network. Original Fuzzy ArtMap specification requires that capacity of network, in terms of number of categories (locations in our case) that network can learn, need to be fixed prior to learning. Prior fixation, of number of categories that network can learn, means that once RSS patterns of a fixed number of locations(with respect to the capacity of network) are learned by network, then more locations cannot be learned (or incorporated) into that network. This limits the application of original Fuzzy ArtMap in terms of dynamically expanding the location system. We adapt original Fuzzy ArtMap in our implementation such that it do not require capacity of network to be fixed prior to learning and allows network to self-scale itself as new categories (locations) are presented to it. Our implementation of our self-scalable fuzzy ArtMap network is available as open source [20]. Learning dynamics of self-scalable Fuzzy ArtMap are shown in Fig. 6.

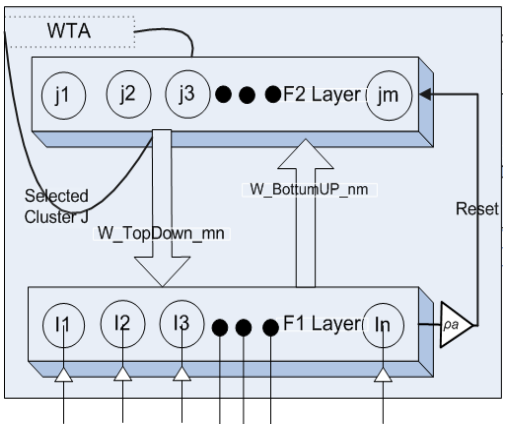


Figure 17: Fuzzy Art Topology

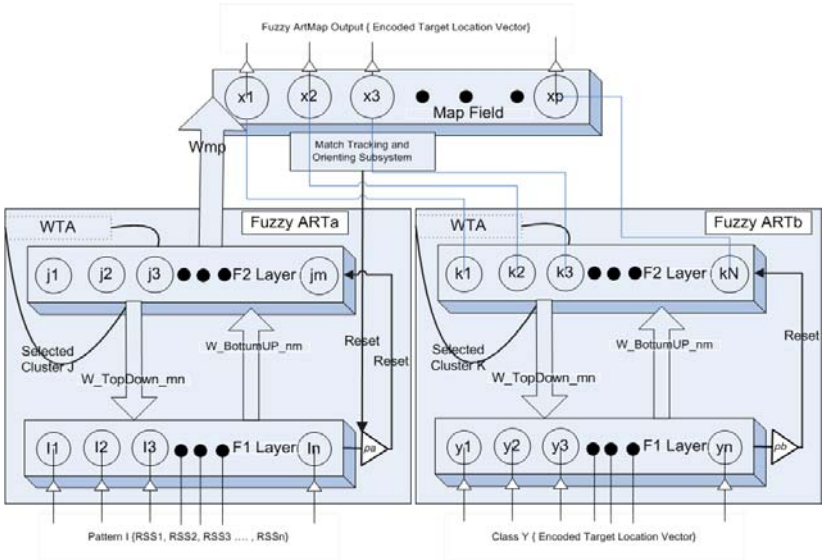


Figure 18: Fuzzy ArtMap Topology

Self Scalable Fuzzy ArtMap

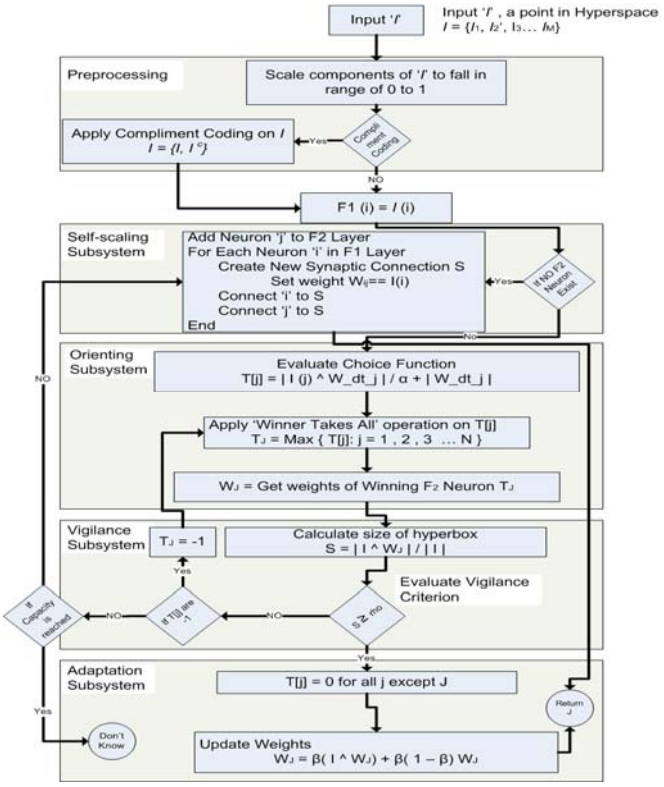


Figure19: Self-scaling Fuzzy Art Learning Algorithm

Fuzzy ArtMap can learn pattern-class pairs online in real time. Online learning of RSS patterns and corresponding locations enables rapid location system development. Fig. 2 (left) shows schematic of development life cycle in o® line training based systems. By virtue of online learning capability of Fuzzy ArtMap this approach removes site calibration phase and o® line training phase from development life cycle, as shown in Fig. 2 (right), and realize rapid location system development. Flexible and Dynamic expansion of location system is easy and straight forward in our approach. By expanding location system flexibly and dynamically we mean incrementally learning new location in real time thus increasing area by including more target locations. In order to achieve this purpose, using previous approaches, Radio Map feature space is required to be extended to include training RSS pattern-location mapping and then retraining of classifier with extended radio map. In case of retraining with new feature space, most of 'online training' based classifiers face with the 'Stability plasticity dilemma'. That means learning new pattern-class mappings causes erosion of previous knowledge acquired by classifier during early training. Other techniques overcome this problem by retraining classifier with whole Radio Map (that includes both old and new training data). Fuzzy ArtMap is capable of incremental learning and ensures stable learning of categories while exposed to new set of pattern-class pairs [4]. This capability allows flexible learning of new locations without requiring retraining with whole new feature space. Learning Rare

Events is very common issue that RSS based location classifiers face. The very nature of radio wave propagation in indoor environments causes imbalanced classes. It means that there may be some locations where size of RSS training patterns is far less than other locations. Fuzzy ArtMap is capable of learning based on a single exposure to an event, and does not show the catastrophic forgetting of rare events that other classification methods do. Moreover inside the building radio wave propagation follows a complex model and produce non stationary feature values in RSS vectors. Fuzzy ArtMap exhibits remarkable ability to classify non stationary data sequences.

Context-Aware Self scalable Fuzzy ArtMap (ConSelFAM)

Due to small scale coverage, there model assumes that all input signals are available at every location all the time. Practically this approach has limited applicability because in real life scenario some signals may not be available at estimation time due to Visibility Problem, as explained in next section. We refer to signal availability of a particular access point at a given location as its 'visibility'. Large scale application of RSS based location estimation faces visibility phenomena. From location estimation stand point, one important aspect of indoor radio wave propagation is that not all access points are visible at all target locations all the time especially in case of large scale location system. We present empirical RSS visibility data in Fig. 1 which shows eight radar graphs of visibility probability of individual Access Points covering experimental site. We identify each access point using last four digits of its MAC (Media Access Code) address. These Access Points are deployed in Department of Computer Engineering building, third Floor, which is shown in Fig. 2 map. As it is obvious from these graphs, every Access Point is visible on a subset of 35 locations. This is because radio signal of certain access point faces attenuation and fading effects and can be accessed within a specific area which is, normally, 150 square meters in indoor environments. Since a particular Access Point constitutes a distinguishing feature for pattern recognition machine, this implies that non-availability of a particular Access Point signal at given location can have adverse affect on location estimation.

By Context-awareness we mean to enable a classifier such that it can differentiate among different input spaces. This is achieved by embedding context knowledge, visibility cluster in our application, into Fuzzy ArtMap classifier. This is achieved by introducing Context Field subsystem into original Fuzzy ArtMap neural network as can be seen in Fig. 20. Learning dynamics, shown in Fig. 21, of Context-aware Fuzzy ArtMap are similar to original one except that it maintains contextual knowledge as a special hash table where context code, visibility cluster in this case, represents key and respective F2 nodes become its values.

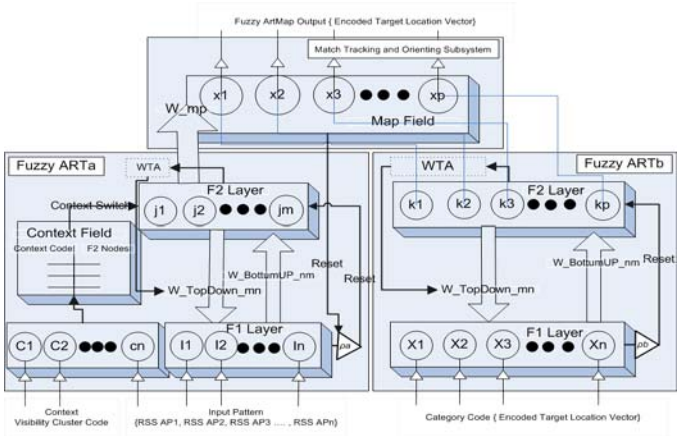


Figure 20: Context-aware, Self-scalable Fuzzy ArtMAP (ConSelFAM)

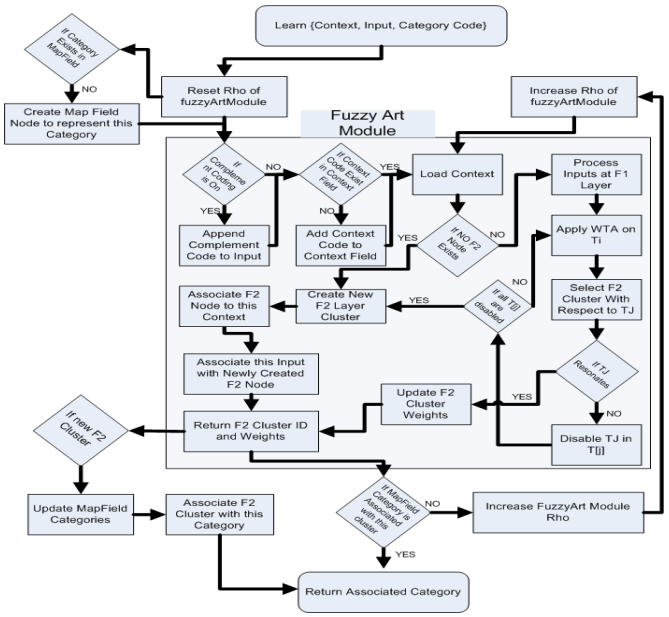


Figure 21: ConSelFAM Learning Algorithm

Radio Map Based Location System

Real Time Learning Machine

We developed a novel approach to develop WLAN based location systems which do not require either a radio map or site specific propagation model while achieving medium scale accuracy. Salient features of this approach are i) Real time learning of the relationship between signal space and physical space which results in lowering the entry barrier for the end users ii) Privacy protection: A mobile device can compute its location in a completely passive manner which enables self-governed privacy protection iii) Unlike enterprise location systems, our approach delivers self-contained location estimation and does not even require network connectivity which is basic assumption in most of previous location systems.

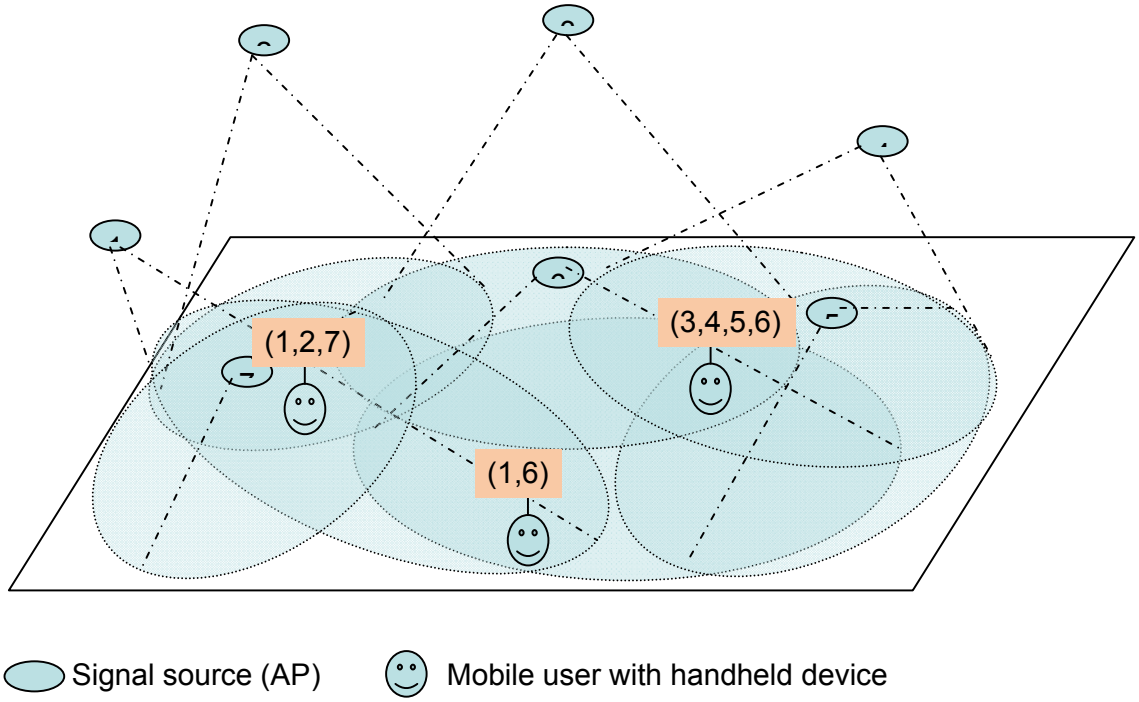


Figure 21: Beacon Based Location Estimation, Example Scenario

This capability achieves Personal Location System concept which functions independent of classical request-response interaction between clients and server. iv) A collaborative development model is realized which enriches the system by a growing community of users. Unlike promised the ease of development by empowering non-professional developers to build, extend and customize complex systems using high level abstractions without learning the underlying technology.

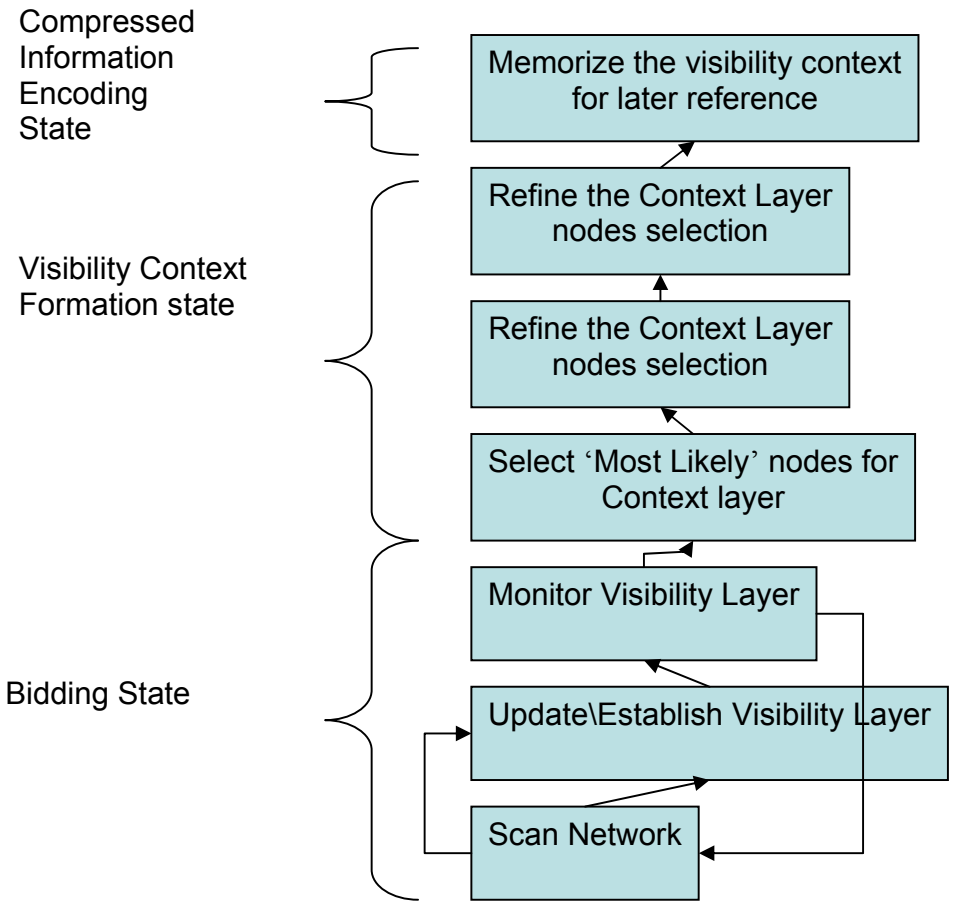


Figure 21: Realtime Learning Algorithm for Dynamic Input Spaces

References

[1] Uzair Ahmad, A. Gavrilov, Sungyoung Lee, Young-Koo Lee," Context-aware Fuzzy ArtMap for Location systems", International Joint Conference on Neural Networks, Florida, USA, August 2007

[2] Uzair Ahmad, A. Gavrilov, Young-Koo Lee, Sungyoung Lee, "Distributed and Parallel Sampling System for Efficient Development of Radio Map", International Conference on Information and Knowledge Engineering, Las Vegas, US, 2007

[3] Uzair Ahmad, A. Gavrilov, Sungyoung Lee, Young-Koo Lee, "Employing Support Vector Machines for Location Estimation", International Conference on Machine Learning: Tools, Methods and Applications, Las Vegas, US, 2007

[4] Uzair Ahmad, A. Gavrilov, Young-Koo Lee, Sungyoung Lee, "Location-ware: A Distributed Wireless Site Calibration System for Location Awareness", International Conference on Information and Knowledge Engineering, Las Vegas, US, 2007

[5] Uzair Ahmad, A. Gavrilov, Sungyoung Lee, Young-Koo Lee, “A Rapid Development Approach for Signal Strength Based Location Estimation", IEEE International Conference on Intelligent Pervasive Computing October 2007, Jeju, Korea.

[6] Uzair Ahmad *et al*, "Modular Multilayer Perceptron for WLAN Based Localization", IEEE International Joint Conference on Neural Networks, July 2006

[7] Uzair Ahmad, A. Gavrilov, Young-Koo Lee, Sungyoung Lee, "In-building Localization using Neural Networks", IEEE International Conference on Engineering of Intelligent Systems, March 2006

[8] Uzair Ahmad, A. Gavrilov, Sungyoung Lee, Young-Koo Lee, “On Building a Reflective Middleware Service for Location-Awareness”. RTCSA 2005: 439-442

[9] Uzair Ahmad, A. Gavrilov, Young-Koo Lee, Sungyoung Lee, “Reflective Middleware for Location-Aware Application Adaptation”. ICCSA (2) 2005: 1045-1054

8.7 Semi-supervised Learning for User-activity Recognition

Semi-supervised learning is a special form of classification. Traditional classifiers use only labeled data (feature / label pairs) to train. Labeled instances however are often difficult, expensive, or time consuming to obtain, as they require the efforts of experienced human annotators. Meanwhile unlabeled data may be relatively easy to collect, but there has been few ways to use them. Semi-supervised learning addresses this problem by using large amount of unlabeled data, together with the labeled data, to build better classifiers. Because
semi-supervised learning requires less human effort and gives higher accuracy, it is of great interest both in theory and in practice.

The popular semi-supervised learning methods include: EM with generative mixture models [1], self-training [2], co-training [3].

Generative mixture models are perhaps the oldest semi-supervised learning method. It assumes a model $p(x, y) = p(y)p(x \mid y)$ where $p(x \mid y)$ is an identifiable mixture distribution, for example Gaussian mixture models. With large amount of unlabeled data, the mixture components can be identified; then ideally we only need one labeled example per component to fully determine the mixture distribution. A few of things should be paid more attend: 1) Identifiability. The mixture model ideally should be identifiable. In general, let $\{p_\theta\}$ be a family of distributions indexed by a parameter vector θ. θ is identifiable if $\theta_1 \neq \theta_2 \Rightarrow p_{\theta_1} \neq p_{\theta_2}$ up to a permutation of mixture components. If the model family is identifiable, in theory with infinite U one can learn θ up to a permutation of component indices. 2) Model correctness. If the mixture model assumption is correct, unlabeled data is guaranteed to improve accuracy. However if the model is wrong, unlabeled data may actually hurt accuracy. It is thus important to carefully construct the mixture model to reflect reality. 3) EM local maxima. Even if the mixture model assumption is correct, in practice mixture components are identified by the Expectation-Maximization (EM) algorithm. EM is prone to local maxima. If a local maximum is far from the global maximum, unlabeled data may again hurt learning. Remedies include smart choice of starting point by active learning. 4) Cluster-and-Label. Instead of using an probabilistic generative mixture model, some approaches employ various clustering algorithms to cluster the whole dataset, then label each cluster with labeled data. Although they can perform well if the particular clustering algorithms match the true data distribution, these approaches are hard to analyze due to their algorithmic nature.

Self-training is a commonly used technique for semi-supervised learning. In self-training a classifier is first trained with the small amount of labeled data. The classifier is then used to classify the unlabeled data. Typically the most confident unlabeled points, together with their predicted labels, are added to the training set. The classifier is re-trained and the procedure repeated. Note the classifier uses its own predictions to teach itself. The procedure is also called self-teaching or bootstrapping (not to be confused with the statistical procedure with the same name). The generative model and EM approach introduced above can be viewed as a special case of 'soft' self-training. One can imagine

that a classification mistake can reinforce itself. Some algorithms try to avoid this by 'unlearn' unlabeled points if the prediction confidence drops below a threshold.

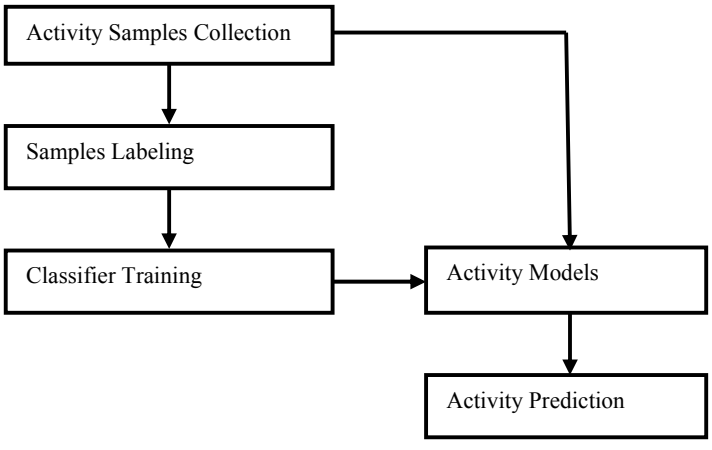
Co-training assumes that features can be split into two sets; Each sub-feature set is sufficient to train a good classifier; The two sets are conditionally independent given the class. Initially two separate classifiers are trained with the labeled data, on the two sub-feature sets respectively. Each classifier then classifies the unlabeled data, and 'teaches' the other classifier with the few unlabeled examples (and the predicted labels) they feel most confident. Each classifier is retrained with the additional training examples given by the other classifier, and the process repeats. In co-training, unlabeled data helps by reducing the version space size. In other words, the two classifiers (or hypotheses) must agree on the much larger unlabeled data as well as the labeled data. We need the assumption that sub-features are sufficiently good, so that we can trust the labels by each learner on U . We need the sub-features to be conditionally independent so that one classifier's high confident data points are *iid* samples for the other classifier.

II. Activity Recognition

The purpose of activity recognition is to infer people's behaviors from low-level data acquired through sensors in a given setting, based on which other critical decisions are made. For example, in smart home environments for aged care monitoring [4][5], based on the information provided by cameras and other pervasive sensors, the system needs to automatically monitor the occupant and determine when they need assistance, raising an alarm if required.

Machine learning is a key aspect in activity recognition. For a system to automatically infer what activity is being performed, it must have a detailed model of the activity. Specially, the system must have the following information: activity data samples got from sensors and labels for the activity samples.

A typical workflow for activity recognition is given in the following graph.



The first step in this figure is to collect activity samples. Samples collection could be characterized by the different usages of sensors. These usages include, (1) remotely

observe the scene using audio, visual, electromagnetic field, or other sensors and interpret the signal readings [6][7][8], (2) attach sensors to the body and interpret the signal readings [9][10][11], (3) attach sensors to objects and devices in the environment and interpret the sensor readings [12][13].

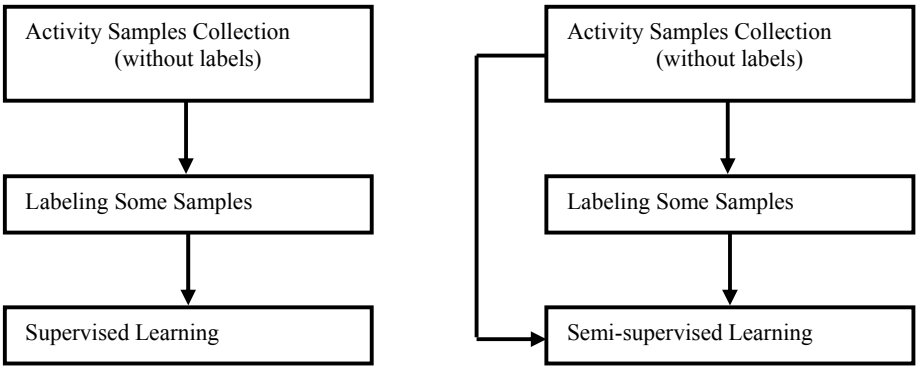
After data collection, some samples are randomly selected and labeled by human. These samples will be used by the classifiers for training purpose. Currently a variety of classifiers have been proposed for activity recognition, such as neural networks [14], dynamic Bayesian networks [15], naïve Bayesian networks [16], hierarchical hidden semi-Markov models [17], nearest neighbors [11], decision tree [11] and so on.

Activity models are generated after classifier training. While the activity is being carried out, data is gathered from sensors. Then the activity data is compared to a set of activity models and inferred which model is the best match.

III. Activity Recognition Based on Semi-supervised learnign

The quality of activity models determines the performance of activity recognition. To achieve good activity models, a large amount of labeled training samples are needed when we use the existing classifiers mentioned above. The reason is that all of them are supervised learning method which could only use labeled data for training.

To reduce human's labeling effort, we propose a semi-supervised learning method for activity recognition. As shown in the following figure, our proposed method can be trained by using both labeled samples and unlabeled samples, hence less labeled samples are needed. Here (a) is for existing methods. (b) is for our proposed method.



(a) (b)

Reference:
1) Nigam, K., McCallum, A. K. Thrun. Text Classification from labeled and unlabeled documents using EM. Machine Learning, 2000 (103-134)
2) Rosenberg, C. Hebert, M. Semi-supervised self-training of object detection models. Seventh IEEE Workshop on Applications of Computer Vision, 2005.
3) Blum, A. and Mitchell, T. Combining labeled and unlabeled data with co-training. COLT: Proceeding of the Workshop on Computational Learning Theory, 1998.

4) V. Stanford: Using pervasive computing to deliver elder care. In IEEE Pervasive Computing, (2002) 10-13
5) C. D. Kidd, R. Orr, G. D. Abowd, C. G. Atkeson, I. A. Essa: The aware home: A living laboratory for ubiquitous computing research. In Proceedings of the Second International Workshop on Cooperative Buildings, (1999) 191-198
6) S.S. Intille, J. Davis, and A. Bobick: Real-time closed-world tracking. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, (1997) 697-703
7) S. Stillman, R. Tanawongsuwan, and I. Essa: A system for tracking and recognizing multiple people with multiple cameras. In Proceedings of the Second International Conference on Audio-Vision-based Person Authentication, 1999.
8) I. Haritaoglu, D. Harwood, and L. Davis: W4: Who, When, Where, What: A real time system for detecting and tracking people. In Proceedings of the Third International Conference on Automatic Face and Gesture, Nara, 1998.
9) M. Makikawa and H. Iizumi: Development of an ambulatory physical activity monitoring device and its application for categorization of actions in daily life. MEDINFO, (1995) 747–750
10) K. Aminian, P. Robert, E. Jequier, and Y. Schutz: Estimation of speed and incline of walking using neural network. In IEEE Transactions on Instrumentation and Measurement, (1995) 743–746
11) L. Bao: Physical activity recognition from acceleration data under seminaturalistic conditions. M.Eng thesis, EECS, Massachusetts Institute of Technology, 2003
12) G.D. Abowd: Director of the AwareHome initiative. Georgia Insitute of Technology, 2002
13) T. Barger, M. Alwan, S. Kell, B. Turner, S. Wood, and A. Naidu: Objective remote assessment of activities of daily living: Analysis of meal preparation patterns. Poster presentation, Medical Automation Research Center, University of Virginia Health System, 2002
14) M. Mozer: The neural network house: an environment that adapts to its inhabitants. In Proceedings of the AAAI Spring Symposium on Intelligent Environments, Technical Report SS-98-02, AAAI Press, Menlo Park, CA, 1998 (110–114)
15) K. P. Murphy: Dynamic Bayesian Networks: Representation, Inference and Learning. PhD thesis, University of California, Berkeley, 2002
16) S.S. Intille and A.F. Bobick: Recognizing planned, multi-person action. Computer Vision and Image Understanding, 2001 (414–445)
17) H. Kautz, O. Etziono, D. Fox, and D. Weld: Foundations of assisted cognition systems. Technical report cse-02-ac-01, University of Washington, Department of Computer Science and Engineering, 2003

9. Security and Privacy Aspects in SCOs

9.1 Introduction

Smart Cooperative Objects (SCOs) have the capability of sensing, computing and communicating with the environment and interacting with other SCOs. They have the functionality of perceiving the environment and making smart decisions based on the context. In short, they are the next generation sensors with the ability to make decisions themselves or by the collaboration of other smart objects in the surroundings. These objects will become ubiquitous in the foreseeable future. As they become more and more pervasive and disappear into the environment, the issue of security and privacy becomes the most important concern.

As a portion in ubiquitous computing (*ubicomp*) network, SCOs pose no novel security issues except what have been introduced in ubicomp. However, providing particular security services for SCOs faces more severe challenges than that of general ubicomp systems. These challenges are collectively inherited from wireless sensor networks, context-awareness systems, mobile computing, and autonomous computing.

Our main focus of this report is on security issues and challenges in SCO environments. We also propose a sketchy security model for smart object. Organization of the report will be as following. In next Section, we first outline the motivation behind our effort of investigation into SCO's security and privacy. Section 9.3 points out the main threats in SCOs. This is the first and indispensable step before we build up security services for a system. In Section 9.4, we investigate thoroughly security issues and challenges in SCO systems. Regarding such issues and challenges, we briefly introduce several related work and dig out what requirements they have and haven't been fulfilled. To our best knowledge, though there have been lots of work done in areas of computer and ubicomp security, there is still no work particularly for SCOs. Within scope of this report, we will take a look at existing approaches and find out whether they are suitable for SCOs. Improvement on those work to apply in SCOs are left for our future work. Section 9.5 provides some future direction and plan. We briefly introduce our current project, *Trust-based Security for Ubiquitous Computing* (TBSI), and propose a revised version of TBSI, *Lightweight TBSI* (μTBSI), aiming to support resource constrains smart object. Finally, Section 9.6 concludes this report.

9.2 Motivation

While SCOs promise to bring comfort to our lives, it might also help the malicious users by giving them new ways to attack the security of the system. There are a number of security, privacy and trust issues that come to our mind when we think of such smart objects becoming ubiquitous. The matter of authenticating and controlling access on such objects when collaborating with personal smart objects, the issue of whether we should trust a certain smart object and the question of privacy preservation are amongst the foremost important concerns. As we try to build a ubiquitous system in which such pervasive smart objects are everywhere scattered in the environment, it is important to tackle the security issues with caution and beforehand. The very features that allow smart

environments to be personalized and dynamic are the features that contribute to the privacy problem. A smart environment will collect data from sensors and from users. The manner of collection will not necessarily be obvious or active. The potential for collection and misuse of information is massive. As pointed out by Campbell [9.1] – this information could be used by the malicious or simply curious, for instance, to track and stalk unsuspecting users. A sentiment echoed in [9.2] when we are reminded of the Orwellian Big Brother Nightmare. Because of this, the demand for privacy is obvious – perhaps even more so than for security or trust. Without notions of individual privacy users will simply not engage with the technology. Research in this area is very important for smart environments where even previous solutions for security and privacy in online systems are inadequate.

9.3 Security Threats in SCOs

Before building necessary security services for a system, it is worth taking a look at the system's threats and how such threats could exploit or damage the system. In SCO environments, there are a number of noticed security threats. Though these threats have been introduced by ubicomp security, they require more concerns in SCOs.

- ***Exposure of User Privacy***: SCOs sense its surrounding context, process and exchange context with other objects. This brings more user privacy concerns such as user location, user activities, etc. For example, a thief can perceive current location and action of the host through a smart object so that he can avoid encountering the host.

- ***Misinterpretation of Context***: The misinterpretation of contextual information also raises problems in SCOs. For example, misinterpretation of temperature of a smart cup can trigger its heater too high so that it could make the holder's hand burned.

- ***Unavailability of Services***: huge services are offered to users through network of SCOs. However, due to erroneous or malicious devices and software, it is likely to make services disrupted. For example, damage of temperature sensors could lead to standstill of the whole heating system.

- ***Compromise of Smart Objects***: SCOs are every day objects such as cups, books, cell phones, etc. This brings more chances for attackers to compromise smart objects and thus no can take over the whole system. For example, an attacker can compromise a smart cup and inject some malicious code into it. This code then automatically executes on the compromised smart object and copies itself to other surrounding smart objects. As a result, those objects are compromised too. The attacker then can easily control the whole system and command the system does whatever he wants through these compromised objects.

- ***Malfunctioning of Autonomous Agent***: autonomous agents facilitate execution on each smart object and collaboration among them. However, the execution of unwanted transactions through autonomous computing agents could bring significant threats to SCO. For example, location exchanging agents on smart objects collaborate

to exchange location information with each other to trigger a certain service (e.g. smart walls automatically display the movie to a user wherever he moves, to kitchen, to dinning room, or to sleeping room). Imagine what could happen if this location context is disclosed to a malicious agent.

9.4 Security Issues and Challenges

Conventional computer systems have introduced three security fundamentals: *Confidentiality*, *Integrity*, and *Availability* (known as CIA). Confidentiality means that sensitive or private information is not disclosed to unauthorized users. In some aspects, it can be considered as privacy matter. Integrity means that information is not modified by unauthorized users. Availability means that resources or services are kept accessible to authorized users.

In order to provide these CIA, ubicomp requires a number of indispensable security services including authentication, access control, security policy, data encryption/decryption and trust collaboration. In this section, we are digging out in more details issues and challenges in each of these subjects.

9.4.1 Authentication

Definition: Authentication is the first process in a secure transaction between two entities. Basically, it is the process of attempting to verify the digital identity of the sender of a communication such as a request to log in. The sender being authenticated may be a person using a computer, a computer itself or a computer program.

As the person moves around, other SCOs might try to identify the personal SCO in order to provide services to the user. If these SCOs are not authenticated, it will result in disclosure of personal information, user tracking and above all malicious smart objects might install malicious code in the personal SCO. The issue of authentication becomes ever more important since we have to take the resource constraints of these objects into consideration. The authentication protocol should also have the additional feature of hiding the user's true identity in some cases.

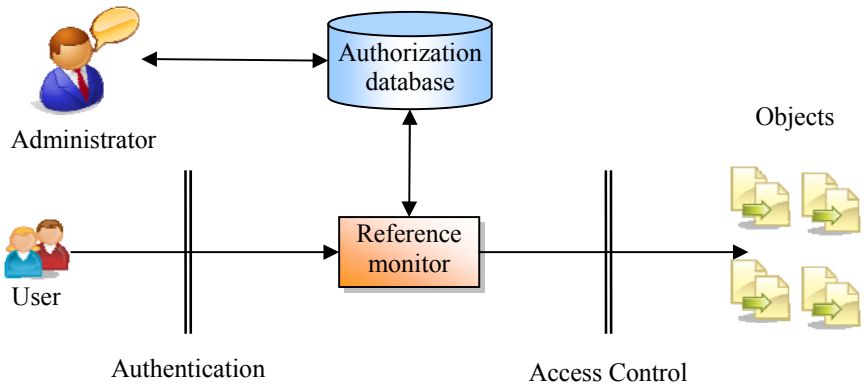
Related Work

Standard cryptographic symmetric and asymmetric techniques can be used to build authentication protocols for low cost pervasive devices like RFIDs. We have certificate based authentication schemes as in SSL, zero knowledge based identification schemes [9.3][9.4][9.5][9.6], using symmetric key techniques like [9.7][9.8][9.9]. There has been much effort in designing protocols which do not rely on standard cryptographic primitives. These protocols are constructed by using lightweight cryptographic methods [9.10][9.11][9.12]. Of note are the protocols HB [9.13] and HB+ [9.14]. HB was designed initially as a candidate identification protocol for human authentication but can be used readily as a light weight protocol for pervasive devices in the form of HB+.

It is also important to maintain privacy at the time of authentication. Pseudonym systems play an important role in this respect [9.15]. There have been efforts to use such systems in the RFID [9.16] setting which preserve privacy while providing authentication.

9.4.2 Access Control

Definition: Access control constrains what a user can do directly, as well as what programs executing on behalf of the users are allowed to do. Access control assumes that authentication of the user has been successfully verified prior to enforcement of access control via a reference monitor. The effectiveness of the access control rests on proper user identification and on the correctness of the authorization governing the reference monitor. Principal of authentication and access control is described in Fig 1.



Authentication and Access Control [2]

It means that, before a user accesses to a certain computer resource or service (so-called object), he must be authenticated first. If his identification is verified, then reference monitor will match his ID into authorization database (in another word, called security policy). If some authorization rule does exist for his identification, then the user can be allowed to perform a certain action (e.g. read, write, etc) on the object, otherwise denied. Conventionally, authorization rules are simple and defined before hand by the system administrator. Emerging of ubicomp raises issues and challenges which are more severe than that.

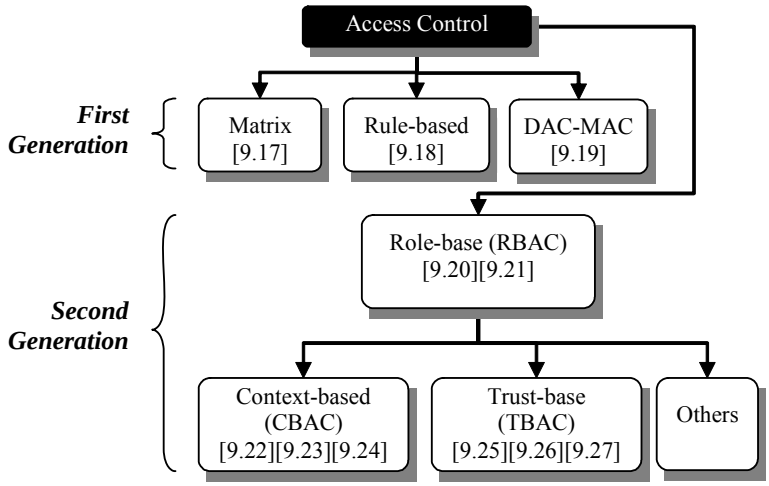
- ***Flexibility***: ubicomp systems take context-awareness as foundation to provide services to users. Conventional approaches are not flexible as such rules (policies) are defined before hand and required manual operation by the administrator to change it. Thus a new model of access control is on call. It must be more flexible, autonomous in the senses that user's access is dynamically controlled based upon the context, the user's *need to know* principal, etc. In SCOs, this is a big challenge because the more flexible the more capacity of operation, memory, and communication are required.

[2] The figure is adapted from "Sandhu R.S *et al.* Access control: principle and practice. IEEE Communications Magazine, 1994"

- ***Scalability***: ubicomp environments consist of thousands to millions smart objects in our daily life. Access control mechanism on each SCO device must be scalable with such a large amount of peer-entities. Traditional approaches are not suitable as they maintain authorization rules based upon individual, not group of individuals.
- ***Uncertainty of entities***: this means that in a certain area (so called *environment domain*), entities (i.e. users, smart objects, etc) enter and leave over time. It is not proper that an entity in that domain maintains all types of authorization rules about every other entity at all the time. This is not possible to store such huge memory on each entity.

Relate Work

In areas of access control technology, we classify into two main generations according to our best knowledge. The first generation, start from 1970s till 1990, involves different types of access control mechanisms which are mainly based on individual identification. The second generation started from early 1990s when the emergence of role-based access control model (RBAC) began. From that on, many access control models have been proposed which applied RBAC as a foundation. A general taxonomy is given in Fig 2 based on our experience.



A General Taxonomy of Access Control

a) First Generation

The earliest work in defining a formal, mathematical description of access control was introduced by Lampson [9.17]. In this model, *subject*, *object*, and an *access matrix* that mediated the access of subjects to objects. An access matrix is a simple conceptual representation in which the (*i,j*) entry in the matrix specifies the rights that subject *i* has to object *j*. In 1973, Bell and Lapadula [9.18] introduced a first rule-based access control model in his paper. Access control rules are formalized into a mathematical model suitable for defining and evaluating computer security systems. As formulated in this model, multilevel secure systems implement the familiar government document

classification rule: users are only allowed to access information that is classified at or below their own clearance level. Conceptually, this is a very simple policy, readily understood and followed by humans. The Bell-LaPadula model was significant because it provided a formal model of the multilevel security policy, making it possible to analyze properties of the model in detail.

In 1983, access control models took a significant step forward when the U.S. Department of Defense (DoD) published its Trusted Computer System Evaluation Criteria (TCSEC) [9.19]. In the standard, two important access control modes were defined: discretionary access control (DAC) and mandatory access control (MAC). DAC is a mode in which the creators or owners of files assign access rights, and a subject with discretionary access to information can pass that information on to another subject.

b) Second Generation

Role-based Access Control: One of the main drawbacks of first generation access control models is that they control access privileges on individual users. In large-scale systems, this will cause high complexity and significant cost to manage. Therefore, a new access control model based on role of users (RBAC) was introduced by D. Ferraiolo [9.20] to tackle the problem. RBAC is conceptually simple: access to computer system objects is based on a user's role in an organization. The authorizations are not assigned directly to particular users, but to roles. A role denotes a job function describing the authority and responsibility conferred on a user assigned to that role.

In 2000, Convington *et al* [9.21] proposes a Generalized Role-Based Access Control model (GRBAC). GRBAC is an extension of the traditional RBAC model for securing application in the highly connected home as well as in other environments. The major benefit of this model is its combination of usability and expressiveness. It solves the problems in RBAC approach by introducing three different kinds of roles: subject roles (e.g. '*adult'*, '*child'*), object roles (e.g. '*image'*, '*source code'*, '*streaming video'*), and environment roles (e.g. '*daytime*', '*nighttime*').

Context-based Access Control (CBAC): Several CBAC models have been proposed aiming to support ubiquitous computing systems [9.22][9.23][9.24]. The underlying technology of ubiquitous computing systems is context-awareness. These approaches take advantages of contextual information such as user location, accessing time, device properties, local resource visibility, etc into controlling access of users. CBAC has become well suitable for such ubiquitous computing environments since this approach adopts context as a design principle to rule access to resources. It dynamically grants and adapts permissions to users according to current context. For instance, if the user is in the presentation room and using the projector, then he will be allowed to use the computer as well without any credential provision because the system recognizes him as a presenter.

Trust-based Access Control (TBAC): Another approach of access control for ubiquitous computing systems is TBAC [9.25][9.26][9.27]. TBAC is the idea of using the human notion of trust and community as a principal concern for assigning privileges.

This means that the user can be granted some access permissions without any prior registration to the system if the user is trusted at some level. TBAC is a promising approach to deal with uncertainty of roaming entities.

Though many types of access control models have been proposed, most of them focus on supporting general ubicomp systems. It brings more challenges to apply those approaches into such resource-constrain devices like smart objects. Therefore, a new, lightweight model is still needed and a trade-off between security level and performance must be thoroughly considered.

9.4.3 Privacy Preservation

Motivation

As already discussed in the introduction to this chapter the very features that allow SCOs to be personalized and dynamic are the features that contribute to the privacy problem. A SCO will collect data from different SCOs and from end-users. The manner of collection will not necessarily be obvious or active. The potential for collection and misuse of information is massive. As pointed out by Campbell [9.28] – this information could be used by the malicious or simply curious, for instance, to track and stalk unsuspecting users. A sentiment echoed in [9.29] when we are reminded of the Orwellian Big Brother Nightmare. Because of this the demand for privacy is obvious – perhaps even more so than for security or trust. Without notions of individual privacy users will simply not engage with the technology. Research in this area is very important for smart environments where even previous solutions for privacy in online systems are inadequate.

The proliferation of action traces (digital data originating from real-world transactions, stored in undesired places), the execution of unwanted transactions through autonomous computing agents, the misinterpretation of contextual information, and service failure due to erroneous or malicious devices and software are amongst the security threats introduced by ubiquitous computing. These threats arise in the interaction of humans with smart objects, within smart environments, where large numbers of smart devices interact, and in conjunction with novel system architectures such as wireless sensor networks, where devices with low resources are employed to monitor mission-critical environmental features.

Definition

According to Alan Westin [9.30] *"privacy is the claim of individuals, groups, or institutions to determine for themselves when, how and to what extent information is communicated to others"*. Privacy on its own is about protecting users' personal information. Considering the advances technology in ubiquitous computing, the concern over privacy is greatly increased. The challenging question, which researchers have begun to tackle recently is how to control and manage users' privacy. Privacy control, as the term states, encompasses the notion of privacy and the notion of control or management. It not only relates to the process of setting rules and enforcing them, but

also to the way privacy is managed/controlled adaptively regarding changes in the degree of disclosure of personal information or user mobility from one SCO to another. The main emphasis is that any good privacy solution should combine these two notions as control is about justification of privacy and plays a role in the management of privacy.

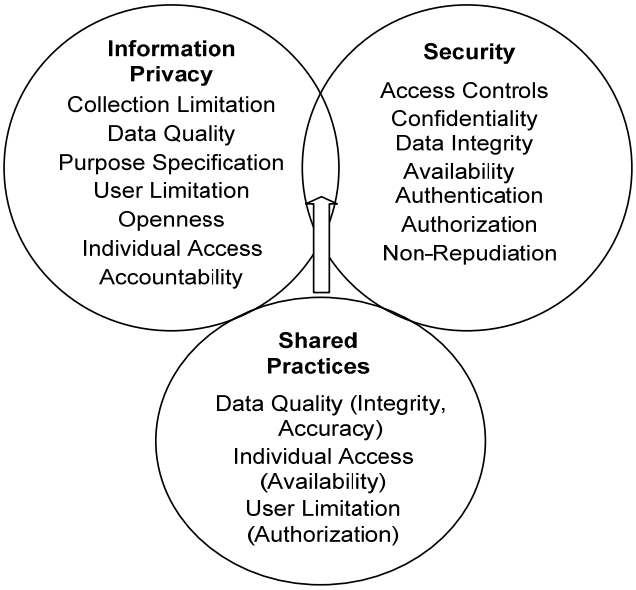


Fig. 3. Security vs Privacy

Our Methodology

In this section, we propose a privacy protection scheme based on the concept of trust with peer recommendation and past interaction history, and the trust-based privacy policy to guarantee that users' privacy sensitive data will not be delivered in a wrong way to a wrongdoer. There are two different stages in our solution: i) we estimate the trust value for each request coming from an entity; ii) we exploit the trust-based privacy policy to make decision how much private data should release to the guest. All these two phases can be performed automatically. We aimed to develop a system that required minimal ongoing user involvement. In particular, we did not want users to have to repeatedly evaluate the acceptability of a request for private information. Instead, we wanted to push a query's acceptance or rejection to the system itself and only bring a query to user's attention if they had not established a policy to handle it. Moreover, we believe user privacy should be protected by default; as a consequence, the system architecture lets a user elect to share certain information rather than protect specific information.

Trust Evaluation

In ubiquitous community, the production of trust is relied on several cues. For example, we tend to trust or distrust potential partners based on their past interactions. We also ask our already trusted principals (e.g. buddy, spouse, supervisor, colleague, secretary, etc, in reality) about their prior experiences with the new prospect uncommon before. The process of the user's system P to evaluate the trust value of any principal Q is shown in Fig. 1.

Time-Based Past Interaction History

Past Interaction History is an entity's previous transaction knowledge to certain principal. As a matter of fact, past interaction history is usually recorded in log files on the subjects' systems that keep track of all actions relational participants took with the system. Since the log file is configured to keep monitoring events for a specified amount of time, it is reasonable for us to apply trust evaluation based on the temporal factor.

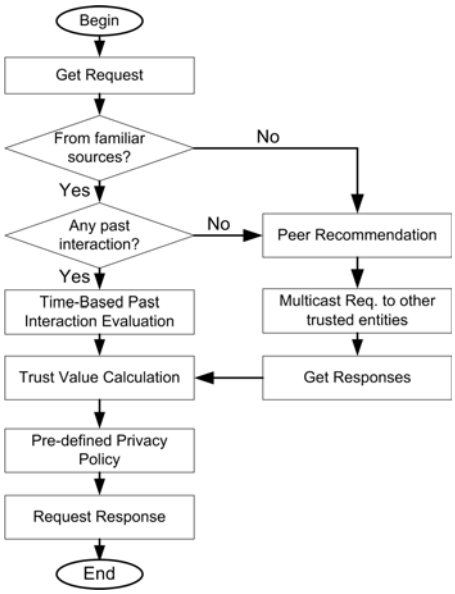


Fig. 1. Flowchart of Trust Evaluation

We can generally define successful and unsuccessful interactions between a principal Q and an application P established on the past behaviors in which an unsuccessful interaction means that the principal did not get the outcome as it expected. Nevertheless, the nature of an interaction might reflect more than just successful and unsuccessful status. For instance, a principal might obtain the result completely contrary to the expectations whereas another one might gain a better effect. Moreover, the outcome of an interaction might be different in the view of the two principals. Due to the complexity of modeling this transition, we restrict our proposed scheme to the two statuses: successful and unsuccessful.

Let us define SI_t as the number of successful past interactions and UI_t as the number of unsuccessful interactions of the system at time t. Now, the trust value of Q as calculated by a system P is defined as follows:

$$T_{P,Q} = 100\left[\frac{SI_t}{SI_t+UI_t}\right]\left[1-\frac{1}{Ae^{(\alpha SI_t-\beta UI_t)}}\right]$$

Where α, β, and A are adjustable positive constants in the system and can be tuned if necessary.

The expression $\left[1-\frac{1}{Ae^{(\alpha SI_t-\beta UI_t)}}\right]$ approaches '1' quickly with an increase in the number of Successful Interactions and/or a decrease in the number of Unsuccessful Interactions within certain period of time. Notice that our choice of the above expression is for the smooth property of the exponential function and ease of calculation. It turns out that $T_{P,Q} = 0$ if $(\alpha SI_t - \beta UI_t) < 0$. In other words, the trust value of principal Q is equal to 0 if its number of Unsuccessful Interactions is greater than the number of Successful Interactions with the system *P*. The factor $\left[\frac{SI_t}{SI_t+UI_t}\right]$ indicates the percentage of successful interactions in the whole communication session. We actually exploit the time-based sliding window mechanism [8] to estimate the percentage of successful communications.

Peer Recommendation

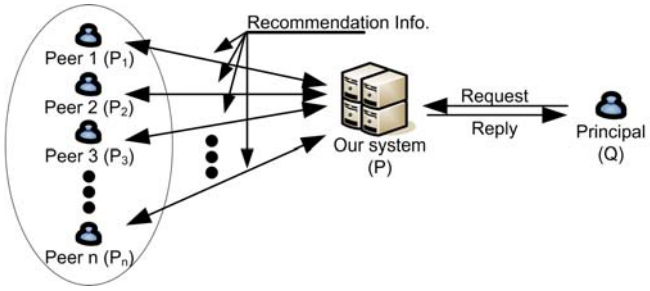


Fig. 3. A Peer Recommendation Scenario

Peer Recommendation factor is required when the system has no or not enough information about a principal. Obviously, if there exists certain peer having more interactions with this principal, his suggestion should be likely logical and important for assessing the trust value. Following the flow chart indicated in Fig. 1, suppose that the system was not familiar with this kind of request before so our system P has to ask other peers in the environment for their suggestions. In this situation, the system will send multicast a request for comments about the new principal Q to its confident community. We denote the time stamp between a principal Q and the system P as $\tau_{P,Q}$ and τ is the time at which Q decides to interact with P. Suppose n is the number of principals currently active in the environment. Let $P_1, P_2, \cdots, P_n$ represent the principals in the space. We also say that principals with high trust values will not send false recommendations. Moreover, let $\Delta\tau$ denote the threshold time interval. Under those assumptions, and Fig. 3, the trust value for the requesting principal Q is defined as follows:

$$T_{P,Q}=\frac{\eta_1 T_{P,P_1} T_{P_1,Q}+\eta_2 T_{P,P_2} T_{P_2,Q}+\eta_3 T_{P,P_3} T_{P_3,Q}+\ldots+\eta_n T_{P,P_n} T_{P_n,Q}}{100n} \quad (n \neq 0)$$

$$\Leftrightarrow T_{P,Q}=\frac{\sum_{i=1}^{n} \eta_i T_{P,P_i} T_{P_i,Q}}{100 \cdot n} \quad (n \neq 0)$$

Where $\eta_i = Be^{\theta\frac{\Delta\tau_{P_i,Q}}{\Delta\tau}} \in (0,1]$, with $\Delta\tau_{P_i,Q} = \tau_{P_i,Q} - \tau$. B and θ are adaptable positive constants which can be chosen apart to guarantee that $\eta_i \leq 1$. For example, we select $\theta = 1$. To

establish $\eta_i \leq 1$, B must be picked out such that $B \in (0, \frac{1}{e^{\frac{\Delta\tau_{P_i,Q}}{\Delta\tau}}}]$. Since $\Delta\tau_{P_i,Q} \leq \Delta\tau$, we have $B_{max} \approx 0.46$. Obviously, $T_{P,Q} = 0$ if $n = 0$. In other words, peer recommendation will not be involved in trust evaluation process if there is no peer in the space. Besides, notice that η_i swiftly approaches '1' with increase in the argument $\Delta\tau_{P_i,Q}$. This means that very old and short experiences of peers with the principal in a period of time $\Delta\tau$ should have less weight in trust estimation over the new and long ones. After finishing the trust evaluation phase, we move towards the second phase in order to decide how much personal data will deliver to the principal (Fig. 1).

Trust-Based Privacy Policy Management

We design a Privacy Policy module to describe the constraints such that the user's data is treated in the manner that she would expect, in the sense of being in accordance with her privacy policy. Once a principal's trust level was quantized by our system, it will be considered as one of three pre-defined states: Trusted, Public or Distrusted with the support of a trust-privacy mapping function $M_P(x)$ as follows:

$$M_P(x) = \begin{cases} \text{Trusted} & , 100 - c_2 \leq x \leq 100 \\ \text{Public} & , 50 - c_1 \leq x < 100 - c_2 \\ \text{Distrusted} & , 0 \leq x < 50 - c_1 \end{cases}$$

Where c_1 and c_2 are adjustable positive constants and can be tuned separately. Respecting this component, we propose 2 different parts, Zone Customization and Privacy Policy Establishment, that help users manage their personal data at the user interface level properly and effectively.

Zone Customization

Inside this sub-module, we develop 3 special zones correlative to 3 distinctive states of a principal Q. Then, we also recommend 3 different privacy control levels for each Trusted & Public Zone. Concerning that point, Public Zone's and Trusted Zone's sliders are used to adjust c_1 and c_2 value in the trust-privacy mapping function respectively.

Privacy Policy Establishment

Whenever the system P receives a request from certain principal Q desiring to query your personal information, it will have to decide whether to place that entity in the Trusted Zone, Public Zone, or Blocked Zone. Placing certain principal in the Trusted Zone enables you to share your privacy-sensitive information and other resources to that principal. Principals you know and get high trust values based on our trust evaluation model should go in the Trusted Zone. Also, placing certain principal in sensitive information to that requester and protects you from the security risks associated with resource sharing. Principals with medium trust values should go in the Public Zone. In the meanwhile, Blocked or Distrusted Zone contains requesters that you do not want to

contact with.

9.4.4 Trust Collaboration

Trust is a complex subject relating to the truster's belief in honest, truthfulness, competence, reliability etc of the trustee. Though trust has been considered extraordinary important in many researches, there is no consensus on what trust is. We use one of the common definitions of trust mentioned in SCO environment:

Definition 1: Trust is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends

The difference between traditional security mechanism (e.g. authentication) and trust was first described in [9.11] as hard security and soft security separately (as shown in Fig.1). Soft security is based on so called social control mechanisms since it is accomplished through the interactions of participants themselves rather than through some central authority when trying to know something about the participants. Hard security on the other hand provides a safe environment and secure communications for SCO. As one of the soft security mechanism, trust system helps build not only service requester' confidence but also a stable environment for service requesters and service providers to carry out transactions with a reduction of risk in SCO environment.

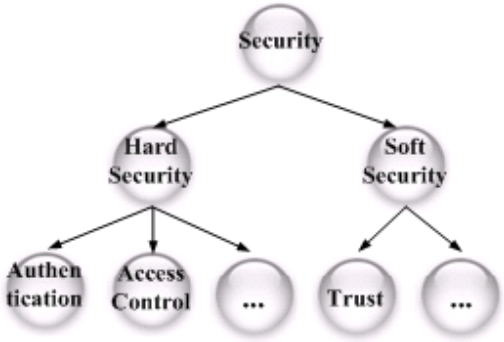


Fig. 4. The Relationship between Security and Trust

When evaluating trustworthiness, trustor's personal observations on the trustee's behaviors are essential. In case personal experience is absent, trust often has to base on recommendations from others. Reputation is a collective measure of trustworthiness based on the recommendations given by other members in a certain community.

Definition 2: Reputation is what is generally said or believed about a person's or thing's character or standing.

The reputations of certain healthcare service providers are based on the ratings given by those who had past interaction history with these service providers in SCO environment.

As smart objects communicate with each other to provide service to end-users, collaboration becomes more and more important. As smart objects are highly dynamic,

enter in and out a domain, they are usually mutual strangers to each other at first. How these smart devices can trust each other so that they can exchange information is nontrivial task(.)

A trust transaction between two entities consists of two parts: trust negotiation and trust management. Trust negotiation established trust between two strangers while trust management maintains all type of information for trust negotiation such as peer-recommendation, history of interaction, etc.

Research in trust area has a long story. Notion of trust at first was applied in economy, then Internet as we have seen nowadays such as PGP (Pretty Good Privacy), SPKI (Simple Public Key Infrastructure). Trust collaboration brings lots of advantage to ubicomp environments, especially SCOs, as most entities are mutual strangers at first. However, several new issues and challenges must be taken into account:

- ***Distinguish between a new object and a known object***: The trust model should be capable of distinguishing between a new object of unknown quality and an object with poor long-term performance.

- ***Awareness of misbehavior***: The trust model should realize and reflect recent trends in the object's performance. For example, a smart object that has acted well for a long time but suddenly goes downhill should be quickly recognized as untrustworthy.

- ***Unfair Ratings:*** In the large-scale, open, dynamic and SCO environments, it is possible that numerous self-interested objects who give unfair ratings to maximize their own gains (perhaps at the cost of others). Therefore, finding ways to avoid or reduce the influence of unfair positive or negative ratings is a fundamental problem.

- ***Free Riders:*** It is essential for the trust model to overcome the free riding behavior. Users who attempt to benefit from the resources of others without offering their own resources in exchange are termed "free-riders".

- ***Changing Identity:*** It is requested that the identity or pseudonym of the objects should be impossible or difficult to change. In case an object has suffered significant loss of reputation it might be in his great interest to change identity or pseudonym in order to cut with the past and start from fresh. However, even though SCO environments enable the objects join dynamically, changing identity is not in the general interests of the community and should be prevented or discouraged.

Related Work

There are some researches that gave helpful attempts on how to get reliable recommendations, e.g. in [9.31], the authors used basic polling as well as enhanced polling. The enhanced polling differs from the basic polling by requesting voters to provide their *servent_id* to prevent a single, possible malicious user to create multiple recommendations at a time. Another very popular method is to give weighted value to

each recommender to choose reliable recommendations. The reputation-based method had been used in a number of works, e.g. weighted majority algorithm is used in [9.32], and a Rating Reputation Feedback mechanism is used to train the weighted values. In [9.33] [9.34] [9.35] [9.36], the authors measure the reputation for each recommender and filter out unfair recommendations based on the usage of the reputation. Using the combination of different filters is also a reasonable method to filter out the unfair recommendations, as mentioned in [9.37][9.38][9.39][9.40]. Their simulation results suggest that cluster filtering is suitable to reduce the effect of unfairly high recommendations and positive discriminations and frequency filtering can guarantee the calculation of trust not be influenced by the unfair raters flooding.

9.4.5 Security Policy

Definition: A *security policy* is a definition of what it means to *be secure* for a system, organization or other entity. For systems, the security policy addresses constraints on functions and flow among them, constraints on access by external systems and adversaries including programs and access to data by people

In *computer security,* policy defines the goals and elements of an organization's computer systems. The definition can be highly formal or informal. Security policies are enforced by organizational policies or security mechanisms.

In *network security,* policy is a generic document that outlines rules for computer network access, determines how policies are enforced and lays out some of the basic architecture of the company security/ network security environment

Though research in security policy specification is not a new subject, but it is more challenging and issued in ubicomp system, especially for smart objects.

- ***A lightweight policy specification***: Each device and smart object within a pervasive environment should be equipped with its own policy database and enforcement mechanism for protecting its own resources. However, SCOs have limited computing power, memory, interfaces, and battery lifetime. Therefore, a lightweight policy model is needed for managing and enforcing security policies.

- ***Semantic and extensible model***: the above limitations and the need for the semantic language to express comprehensive policy, we need a semantic and extensible policy language. The policy engine should be able to dynamically decide what rights, prohibitions, obligations, dispensations an actor has on the domain actions.

- ***Effective policy for diverse interaction***: In ubicomp, real-world actions are seamlessly reflected by transactions in the virtual space. In such environments, security policies are important both in the physical and the logical world. In smart environment, interactions between users and smart objects, and especially, mobility of users and smart objects tie logical and physical security together. We need to design an effective policy model and policy specification to afford this requirement

Related Work

The Resurrecting Duckling security policy model [9.41] describes a lightweight way of establishing a secure transient association between two devices - master device and slave device. The extended version of this policy model even encompasses a great variety of relationships between devices. We deal with peer-to-pear interactions, or have some levels of being master.

Lalana Kagal *et al.* [9.42] describe Rei - A Policy Language for Pervasive Computing Environment. It is a lightweight policy language, including constructs for rights, prohibitions, obligations and dispensations (deferred obligations). The language consists of a few simple constructions that are extremely flexible and allows different kinds of policies (security, privacy, management, conversation etc.) to be specified. The policy language is not tied to any specific application and permits domain specific information to be added without modification. Rei models speech acts like delegation, revocation, request and cancel that allow policies to be less exhaustive and allow for decentralized security control.

Anand *et al.* [9.43] presents an architecture and a proof of concept implementation of a security infrastructure for mobile devices in an infrastructure based pervasive environment. It consists of two parts, the policy engine and the policy enforcement mechanism. Each mobile device within a pervasive environment is equipped with its own policy enforcement mechanism and is responsible for protecting its resources. The system wide policy is described in a semantic language “Rei”, a lightweight and extensible language which is able to express comprehensive policies using domain specific information. It is considerable to apply for this architecture to SCOs in ubiquitous environment.

Security Policy Model and Policy Types

Security policy model bases on access control model. We treat policies as objects in security policy model, which has common elements as in policy management framework: enforcement module, policy management tool (UI interface), policy repository, policy manager, policy domain.

Note that some access control models use risk, trust, and context as important components. Access control is concerned with limiting the activity of legitimate users. In this work, security policy is mostly access control policy. Our scope is about nondiscretionary access control where administrators have the authority to specify security policies that are enforced by the access control system. Delegation and propagation of authority are permitted only within the scope defined by the security policy. We focus on authorization, delegation, and information policies.

Authorization Policies

Authorization policies define what activities a member of the subject domain can perform on the set of objects in the target domain in terms of interface method calls. A positive authorization policy defines the actions that subjects are permitted to perform on target objects. A negative authorization policy specifies the actions that subjects are forbidden to perform on target objects. Authorization policies are implemented on the target host by an access control enforcement component (an access controller), traditionally called a reference monitor. Constraints are optional in all types of policies and can be specified to limit the applicability of policies based on time or values of the attributes of the objects to which the policy refers. Note that the subject and target elements can optionally include the interface specification reference within the specified domain-scope-expression on which the policy applies. This can be used to check that the objects do support the specified operations or to locate the interface specification. It can be argued that the specification of negative authorization policies complicates the enforcement of authorization in a system. However, there are reasons to support the provision for negative authorization policies. Administrators often express high-level access control in terms of both positive and negative policies; retaining the natural way people express policies is important and provides greater flexibility. Negative authorization policies can also be used to temporarily remove access rights from subjects if the need arises. In addition, many systems support negative access rights (e.g., Windows NT/2000). The existence of both positive and negative authorization policies in a system may result in conflicts. Although this adds the need to analyze policies for conflict detection, this kind of conflict may indicate potentially unforeseen problems with the specification.

Delegation Policies

Delegation is often used in access control systems to cater for the temporary transfer of access rights. However the ability of a user to delegate access rights to another must be tightly controlled by security policies. This requirement is critical in systems allowing cascaded delegation of access rights. A delegation policy permits subjects to grant privileges, which they possess (due to an existing authorization policy), to other subjects called grantees to perform an action on their behalf e.g., passing read rights to a printer spooler in order to print a file. A delegation policy is always associated with an authorization policy, which specifies the access rights that can be delegated. Negative delegation policies forbid delegation of certain actions. Note that delegation policies are not meant to be used for assignment of rights by security administrators.

Information Filtering

Filtering policies are needed to transform the values of the input parameters in an action and the information returned from the action. For example, a location service might only permit access to detailed location information, such as a person is in a specific room, to users within the department. External users can only determine whether a person is at work or not. Some databases support similar concepts of ‘views’ onto selective information within records – for example a payroll clerk is only permitted to read personnel records of employees below a particular grade. Positive authorization policies may include filters to transform input parameters associated with their actions, based on

attributes of the subject or target or on system parameters (e.g., time). In many cases it is not practical to provide different operations as a means of selecting the information. Although these are a form of authorization policy they differ from the normal ones in that it is not possible for an external authorization agent to make an access control decision based on whether or not an operation, specified at the interface to the target object, is permitted. Essentially the operation has to be performed and then a decision made on whether to allow results to be returned to the subject or whether the results need to be transformed. Filters can only be applied to positive authorization actions. Every action can be associated with a number of filter expressions. Each filter contains an optional condition under which the filter is applied. If the condition evaluates to true, then the transformations (the assignment statements in the body of the filter) are executed.

9.4.6 Intrusion Detection System

Introduction

Intrusion detection (ID) is defined as "the problem of identifying individuals who are using a computer system without authorization and those who have legitimate access to the system but are abusing their privileges". The intrusion detection field has grown considerably in the last few years, and a large number of intrusion detection systems have been developed to address different needs. Intrusion detection is clearly more necessary in ubiquitous networks where other secure systems such as firewall are not applicable. In this Smart Spaces scenario, we focus on designing an IDS architecture for the network of smart objects.

Networks of smart objects constitute a new paradigm of ambient monitoring with many potential applications. Typically formed by thousand of nodes of small dimension, they use ad-hoc communication and have scarce resources regarding energy, bandwidth, processing capacity and storage.

Smart objects are typically designed to gather data in inhospitable places and might be involved in critical applications. Wealth environment mapping and enemy's movement monitoring in a battlefield are some examples of critical applications they are used for. In these applications, Smart objects are of interest to adversaries. Smart objects are susceptible to some types of attacks since they are deployed in open and unprotected environments and are constituted of cheap small devices. Preventive mechanisms can be applied to protect Smart objects against some types of attacks. However, there are some attacks for which there is no known prevention method, such as wormhole. Moreover, there are no guarantees that the preventive methods will be able to hold the intruders. For these cases, it is necessary to use some mechanism of intrusion detection. Besides preventing the intruder from causing damages to the network, the intrusion detection system (IDS) can acquire information related to the attack techniques, helping in the development of prevention systems.

Intrusion detection poses many challenges to Smart objects, mainly due to the lack of resources. Besides, methods developed to be used in traditional networks cannot be applied directly to Smart objects, since they demand resources not available in networks of smart objects. Smart objects are typically application-oriented, which means they are designed to have very specific characteristics according to the target application. The intrusion detection assumes that the normal system behavior is different from the behavior of a system under attack. The several possible WSN configurations make difficult the definition of the "usual" or "expected" system

behavior. Since common nodes are designed to be cheap and small, they do not have enough hardware resources. Thus, the available memory may not be sufficient to create a detection log file. Moreover, a smart object is designed to be disposed after being used by the application and it makes difficult to recover a log file due to the possible dangerous environment in which the network was deployed. The software stored in the node must be designed to save as much energy as possible in order to extend the network lifetime. Finally, another challenge to the design of an IDS is the frequent failures of smart objects when compared to processing entities found in wired networks. Given all these characteristics, it is important to detect the intrusions in real time. In this way, we could hold the intruder and minimize the application damages.

Architecture

The architecture takes full advantage of the secure & powerful server in Smart Space scenario. It includes two parts: ID server and ID agents. Most of the intrusion detection processing is done in the server side to reduce processing amount in the client side. ID agents are installed in every smart objects to collect data and detect anomaly behaviors.

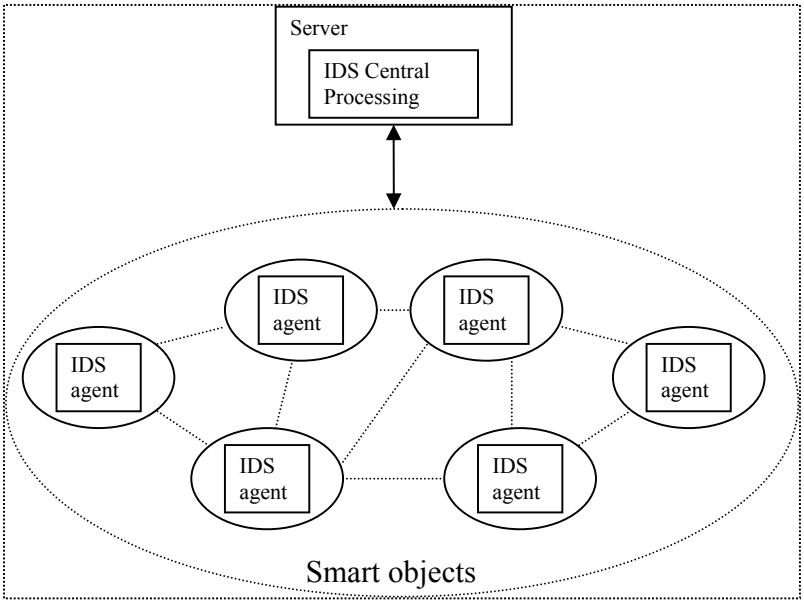


Fig 9.1 Intrusion Detection Architecture

ID Agents

Because networks of smart objects lack of central point to collect data so there is no doubt that every intrusion detection architecture must install detection agents on many nodes in order to cover the entire network. The important thing is how agents should do to save energy & other resources of the host nodes, however, still guarantee the security for the host nodes & all network.

ID agent is responsible for monitoring the host node & its neighbors for anomaly behaviors. Because of limited resources of smart objects, ID agent must be as lightweight as possible. In our architecture, agents use some simple rules to detect intrusion.

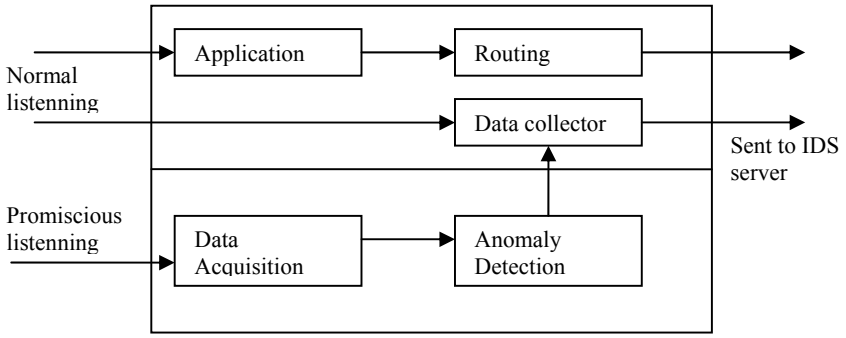


Fig 9.2 IDS agent architecture

Data Acquisition

In this phase, messages are listened to in promiscuous mode by the monitor mode and the important information is filtered and stored for subsequent analysis. Important information includes message fields that might be useful to the rule application phase. Thus, we use less memory and less processing time, saving energy. Messages to which no rules can be applied are not stored.

Data extracted from the messages are stored in an array data structure and discarded after a given period of time or when there is no space left in memory.

Anomaly Detection:

This module detects anomaly behavior in neighboring nodes based on statistical data of packets received by promiscuous listening. This module supplements Rule Application module.

Data Collector Module:

This module is in charge of collecting & sending necessary data to the IDS server for father processing. However, in the sake of saving resource, this module is active only when the IDS agent detects something abnormal in the network. IDS agents rarely go to the final decision of intrusion detection. This is in charge of IDS server.

IDS Server

IDS server is the central processing in this architecture. IDS server will collect all data from necessary smart objects, aggregating & making the final intrusion decision. IDS server can use both signature based and anomaly detection technique.

9.4.7 Smart Networking Sensors

As we are aware of the fact the wireless sensor networks usually consist of tiny sized sensor nodes that are densely deployed in a monitoring environment. The basic purpose of sensor nodes is to sense, gather and propagate information about events that occurs within monitoring environment. But when we associate a 'smart' terminology with sensor networks, it means that sensor nodes have now some additional capabilities of intelligent processing and/or context awareness capability [9.44]. It also means that a smart sensor network offers "powerful capability to locally and collaboratively sense both personal and environmental data, reason and interpret collected data, and react to various situations" [9.45].

All the security threats of sensor networks are equally but with more intensity applicable to smart sensor networks. In general, there are number of different threats to the sensor networks like DoS, eavesdropping, message injection, message replay, message modification, malicious code and side channel analysis etc. The security primitives against these attacks are message confidentiality, authentication, service availability, message freshness, message integrity, non-repudiation, intrusion detection, and audit trials etc. In a smart wireless sensor network, it is much easier to monitor transmission between nodes as compared to wired networks because of the broadcasting nature of transmission. Encrypting communication between sensor nodes can partly solve this problem but it requires robust key exchange and distribution scheme, compelling the wireless sensor networks to maintain secrecy in the rest of the network when an adversary compromises few sensor nodes and exposes their secret keys.

9.4.8 Key management

Definition: *Key management* is the fundamental issue for securing sensor networks. Its purporse is guarantee the shared secret key between entities, which is the basis for security solutions against threats in sensor network such as passive eavedropping, active message modification, denial of service, object fabrication…

Smart objects are devices with sensing, computation, and communication capabilities and are able to perceive and interact with their environment and with other smart objects. Because of the co-operating to sharing information over wireless communication in the network and the resource constraint, designing a flexible and efficient key management scheme for smart objects network is not a trivial task.

Key management schemes on SCO network have issues and challenges as follow:

- **Scalability**: is the ability to support different size of network and flexible in the changing size of network even after deployment.
- **Efficiency** in using resource (storage, processing, and communication): because SCOs is the limited-resource devices, so the key management solutions must be lightweight schemes.

- **Key connectivity**: is the capability of two entities in the network could have the same key or keying material. Enough key connectivity provides intended services of system.
- **Resilience against objects captured**: due to the facts that SCOs are everyday objects, so they're easy to be compromised by attackers. Key management schemes have challenges with this issue in order to guarantee secure communication in system even though some objects are stolen.

Related works

In sensor network, there are several approaches on key distribution and management. The first group is using public key cryptography and elliptic key cryptography to adapt low-powers devices, such approaches is considered as costly due to high processing requirement [9.45][9.46]. SCOs also have limited computation, energy… like sensor node. So that the symmetric key cryptography are the feasible approaches.

Solutions to key distribution using symmetric cryptography in distributed sensor network can use one of the three approaches: (i) probabilistic, (ii) deterministic, (iii) hybrid. Probabilistic key distribution solutions base on the idea that key rings are built from a key-pool and distribute to sensor nodes. The random graph show that any two nodes could establish a secure connection based on sharing key with specific probability. The deterministic solutions using deterministic processes to design the key-pool and key-chain to provide better connectivity. And the hybrid solutions use probabilistic approaches on deterministic mechanisms to improve the performance and security.

The first scheme in probabilistic key distribution solutions is proposed by Eschenaueur and Gligor [9.47]. In this scheme, a large key pool is generated off-line and every sensor picks a random subset of keys from this key pool. Any two nodes in the communication range can talk with each other only if the share a common key. Depending on the size of the key pool and the number of sensor nodes in the network, the scheme could archives different connectivity and resilience. Chan, Perrig and Song [9.48] latter proposed to use the same idea, but increase the intersection sharing keys between key-rings from one key to some $q>1$ keys. It is shown that, by increasing the value of q, network resilience against node capture is improved. Further, Du et al. proposed key predistribution by using deployment knowledge [9.49], that each group sharing α keys with horizontal groups and β keys with diagonal groups. Because the number of object in SCOs network is quite less than the number of nodes in wireless sensor network, the requirement that any object must be connected, whereas probability key distribution solutions only support connectivity with percentage less than 100%. Therefore, probabilistic key distribution is not suitable for SCOs network.

To guarantee better connectivity, deterministic mechanisms such as polynomial-based schemes and key matrix-based schemes are proposed. Polynomial scheme of Blundo et al. [9.50] using polynomial evaluations to obtain a pairwise key. Each node i gets polynomial share f(i,x) of symmetric polynomials of degree t. In order to calculate the common key with node j, node i needs to evaluate its polynomial with x=j: f(i, x=j). Node

j would evaluate f(j, x=i) and because of symmetric polynomials, they have f(i,j)=f(j,i). So two nodes calculate the common value, then use as sharing key. This scheme is resistant against node capture with property *t-secure*: compromise of less than t+1 node doesn't reveal any information about keys of other nodes. Applied to sensor network, there are schemes using sensor's predeployment knowledge such as grid-based key predistribution of Liu and Ning [9.51,9.52], hexagon-based key predistribution of Zhou et al.[9.53], and t-neighborhood scheme of Das et al.[9.54]. The hybrid approach, which was combined from Blundo's scheme and the random key pre-distribution of Eschenauer-Gligor, is *random-subset assigment key predistribution* of Liu and Ning [9.55]. Instead generating large key-pools and create key-rings, this scheme generated a large polynomials and assigned each node a subset of polynomials from the pool. Then two nodes can only communicate when they sharing at least one polynomials. It's shown that this scheme increased the resilience when comparing with Eschenauer-Gligor's scheme.

The key matrix mechanism was firstly introduced in [9.56]. Blom proposed a key predistribution scheme that guarantees any pair of members in a group could calculate the common sharing key. He introduced symmetric key matrix K, where element $K_{i,j}=K_{j,i}$ is a pairwise key between node i and node j. The key matrix K is obtained by multiplication of public matrix G which is known by all nodes and private matrix D. Each node i stores a corresponding row of private matrix D and is able to compute a pairwise key with any node j by multiplying the row it stores with j-th column in public matrix G. This is the basic for further schemes applied to sensor network. Du et al. combined the Blom method with random predistribution scheme [9.57]. Calling the set of keys that each tuple <D,G> can generate is a key space, scheme in [9.57] selected randomly τ spaces to each node from ω pre-generated spaces. Based on probabilistic, any two nodes could share a common space, computing a common secret key. Latter, Du et al. also applied predeployment knowledge to *DDHV-D scheme* in [9.58]. It's the hybrid solution combined from key space and random sharing key scheme in [9.49].

In hierarchical network topology, unicast communications are between base station and cluster heads, between cluster heads and smart objects. Solution for such situations is straightforward; base station can share a distinct pair-wise key with each object. Very similar solutions are proposed in *Perimeter protection scenario*[9.59], *Base station authentication protocols* [9.60][9.61], *Localized encryption and authentication protocol (LEAP)*[9.62]. *LEAP* also supports multicast communication using group-wise key. In broadcast which require network-wise key, such solutions are *Time Efficient Stream Loss-tolerant Authentication (TESLA)*[9.63], µ-TESLA [9.64] and extended versions of µ-TESLA [9.65][9.66].

Dynamic key management models with rekeying mechanism are offered in [9.67][9.68]. The exclusion-based systems (EBSs), a combinatorial formulation of the group key management, firsly used in [9.67] to takle the problem of dynamic rekeying. In EBS-based schemes, each node is assigned k keys out of a pool of size P = k+m keys. Rekeying takes place either periodically or once one or more nodes are captured (or suspected of being captured). Replacement keys are generated, then encrypted with all

the m keys unknown to the captured nodes, and finally distributed to other nodes that collectively know the m keys. Rekeying takes place either periodically or once one or more nodes are captured (or suspected of being captured). Replacement keys are generated, then encrypted with all the m keys unknown to the captured nodes, and finally distributed to other nodes that collectively know the m keys. A drawback of the basic EBS-based solution is that a small number of nodes may collude and collectively reveal all the network keys. This is particularly true when the value of m is selected to be relatively small (to make rekeying feasible in terms of number of messages). In order to address the collusion problem in EBS, Younis et al proposed SHELL [9.68], an EBS-based scheme that performs locationbased key assignment to minimize the number of keys revealed by capturing collocated nodes. SHELL uses the EBS framework to perform rekeying within each cluster. Cluster gateways keep track of the key assignment but not the actual keys. These keys (k keys assigned to each node) are stored by gateways of other clusters (called key generation gateways). Keys are distributed to nodes by the key generation gateways through their own cluster gateway using an extra cycle of encryption/ decryption. SHELL is collusion-resistant. This model relies considerably on a centralized key server to perform rekeying.

9.5 Future Direction and Plan

9.5.1 Current Research: Trust-based Security Infrastructure (TBSI) Project

Currently, we are working on Trust-based Security Infrastructure for Ubicomp Systems (TBSI) project. This project aims to support basic yet important security services for ubiquitous computing environments, such as smart house, smart office, pervasive healthcare etc. TBSI architecture is described in Fig 4. Fig 4 shows its diagram.

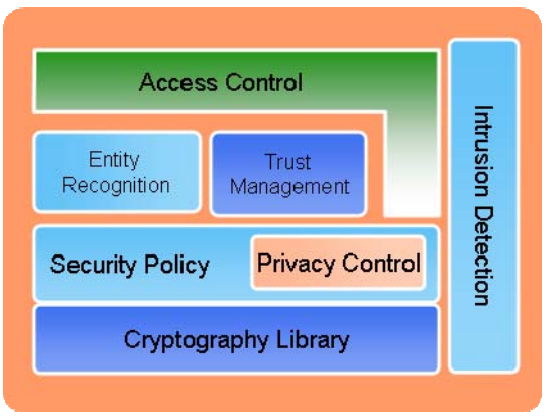


Fig. 7. Trust-based Security Infrastructure (TBSI) Architecture

TBSI includes six components: Cryptography Library, Security Policy (including Privacy Control), Trust Management, Authentication, Access Control, and Intrusion Detection.

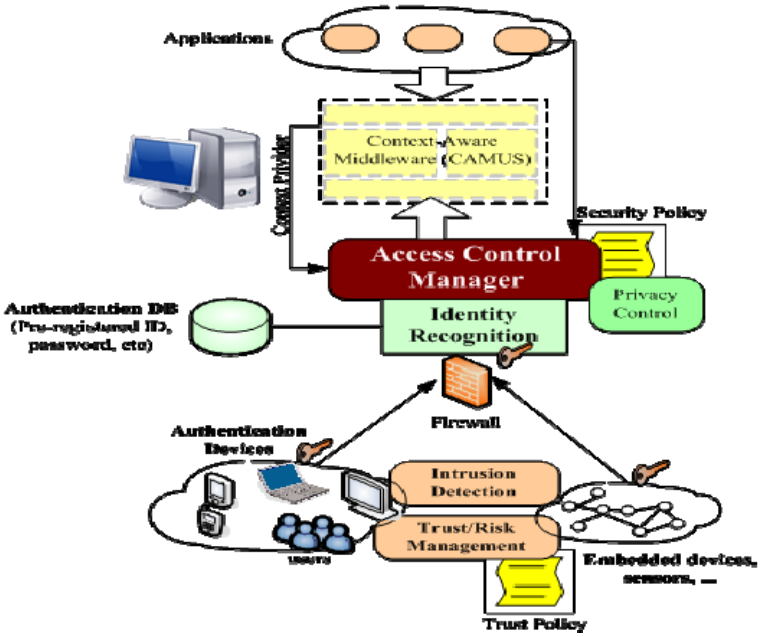


Fig. 8. TBSI diagram

Entity Recognition is a new authentication technology for ubiquitous computing paradigm. In TBSI architecture, entity recognition supports various flexible devices such as Smart Badges, iButtons, Smart Watchs, PDAs. This component integrates different type of authentications, ranging from conventional authentication approaches (Username/Password, PKI, Kerberos, etc) to emerging identity recognition technologies.

Trust Management provides trust value to the Access Control Manager. It supports trust collaborations and interactions which usually occur among roaming entities. By modeling trust relationships in smart spaces environments, unknown entities from different domains can interact, request services and resources from a given domain in secure and privacy manner.

Access Control is the core part of TBSI infrastructure. Our Access Control model utilizes trust and context to solve different shortcomings of existing role-based access control (RBAC), context-based access control (CBAC), and trust-based access control (TBAC). It is critical to preserve confidentiality and integrity. Conventionally, the condition of confidentiality requires that only authorized users can read information, and the condition of integrity requires that only authorized users can alter information in authorized ways. Access control extends scopes of users by using Trust Management. By doing this, it supports not only predefined entities but also un-predefined entities.

Privacy Control is an integral part in this convenient but obtrusive environment. It provides *location privacy, anonymous connections* and *confidentiality* of information to users. In TBSI, Intrusion Detection System is deployed in order to defend against unauthorized access and who has legitimate access to the system but abuse privileges. In ubiquitous environments, this usually occurs due to ubiquity and wireless communication of the systems. In the sensor network layer, TBSI provides a lightweight cryptography

mechanism in order to maintain secure communication among sensors and between sensors and context-aware systems. Trust Management, Intrusion Detection System, Home Firewall, and Sensor Network Security together support Entity Recognition.

9.5.2 Lightweight TBSI (*μTBSI*) for SCOs

Due to aforementioned issues and challenges, we aim to provide a lightweight security services for SCOs. Based on our TBSI project, we propose a *μTBSI* (Lightweight Trust-based Security Architecture) targeting to support low resource smart objects. *μTBSI* is integrated with basic yet crucial security services: cryptography library, security policy (including privacy control), entity recognition, trust management, and access control. The simplified architecture is depicted in Fig 5.

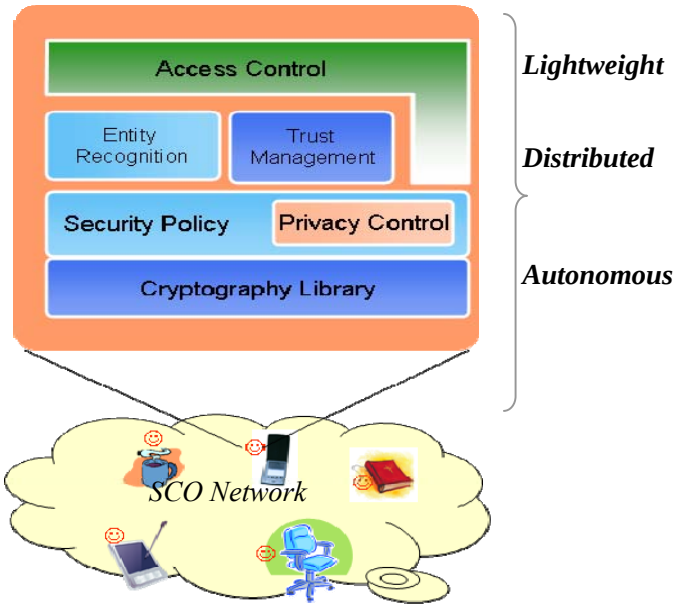


Fig. 9. Lightweight Trust-based Security Infrastructure (*μTBSI*) for Smart Cooperative Objects

In order to provide efficient security services for SCOs, *μTBSI* must be satisfied with the following requirements:

a. ***Lightweight***: with low energy, computation and communication capacity, each source code must be small enough to make it possible and efficient to run on each object.
b. ***Distributed***: it's not possible to maintain all above security services in a smart object at once. Distribution of services in secure manner must be taken into consideration.
c. ***Autonomous***: due to various and numerous small devices, management of each SCO is high cost and complex. Providing an autonomous mechanism for each security services is therefore necessary.

9.5.3 Outline of 5-Years Plan

- ✓ **1st year: Survey on**
  - Smart Cooperative Objects scenarios and applications
  - Aspects of security for networking sensors (including encryption/decryption algorithms, key management schemes, authentication, trust management, privacy control, access control, etc)

- ✓ **2nd year: Modeling and Design**
  - Focus on *μTBSI* architecture modeling and design. Provide detailed architecture of each security services and API between them.

- ✓ **3rd year: Implementation**
  - Implement *μTBSI* by Java and C.

- ✓ **4th year: Test-bed**
  - Running *μTBSI* code by either simulation or test-bed.
  - Making performance and evaluation
  - Revise and utilize the source code

- ✓ **5th year: Deploy on Real Application**
  - Customize and deploy *μTBSI* on real applications. Possible applications could be smart office, ubiquitous healthcare.
  - Wrap the product
  - Making manual and related documentations

9.6 Conclusion

In this report, our most effort is to investigate security issues and challenges in smart cooperative object (SCO) environment. After reviewing traditional approaches in computer security as well as existing approaches applied to ubicomp systems, we come to know that there is no new security issue in SCOs except what have been introduced into ubicomp systems. Nevertheless, in order to tackle those issues, it is much more challenging. Those challenges are collectively inherited from context-awareness security, wireless sensor network security, mobile computing security, and autonomous computing security. We then focus on the most important issues that must be thoroughly considered for SCOs. We also take a look at what research community has been doing so far and whether existing security approaches can be applied in SCO environments. We then propose a sketch μTBSI (a Lightweight Trust-based Security Infrastructure) for SCOs. μTBSI is the revised version of our current project TBSI (Trust-based Security Infrastructure for Ubiquitous Computing).

References

[9.1] Roy Campbell, Jalal Al-Muhtadi, Prasad Naldurg, Geetanjali Sampemane, M. Dennis Mickunas, “Towards Security and Privacy for Pervasive Computing.” in Theories and Systems, Mext-NSF-JSPS International Sympsoium, ISSS 2002, Tokyo, Japan, November 2002. pp. 1-15,

[9.2] Marc Langeheinrich, “Privacy by Design : Principles of Privacy Aware Ubiquitous Systems,” Proceedings of the Third International Conference on Ubiquitous Computing (UbiComp 2001). LNCS No. 2201, Springer-Verlag, pp. 273--291, Atlanta, USA, 2001.

[9.3] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Advances in Cryptology - CRYPTO’86, Lecture Notes in Computer Science 263, p.p. 186–194. Springer-Verlag, Berlin, 1987

[9.4] Uriel Feige, Amos Fiat, and Adi Shamir. Zero knowledge proofs of identity. In Proc. the 9th annual ACM conference on Theory of computing (STOC’87), pages 210–217. ACM Press New York, 1987.

[9.5] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In Advances in Cryptology - EUROCRYPT’88, LNCS No. 330, p.p. 123–128.1988.

[9.6] C.P. Schnorr. Efficient identification and signatures for smart cards. In Advances in Cryptology - CRYPTO’89, LNCS No. 435, p.p. 239–252. 1990

[9.7] Henrici, D., and M¨uller, P. Hash-based Enhancement of Location Privacy for Radio-Frequency Identification Devices using Varying Identifiers. In Pervasive Computing and Communications (PerCom) (2004), IEEE Computer Society, pp. 149–153.

[9.8] Molnar, D., and Wagner, D. Privacy and Security in Library RFID : Issues, Practices, and Architectures. In Computer and Communications Security (2004), B. Pfitzmann and P. McDaniel, Eds., ACM, pp. 210 – 219.

[9.9] Ohkubo, M., Suzuki, K., and Kinoshita, S. Efficient Hash-Chain Based RFID Privacy Protection Scheme. In Ubicomp (UBICOMP) (September 2004).

[9.10] Ari Juels. Minimalist cryptography for low-cost RFID tags. In Carlo Blundo and Stelvio Cimato, editors, The Fourth International Conference on Security in Communication Networks - SCN 2004, volume 3352 of Lecture Notes in Computer Science, pages 149--164, Amal, Italia, September 2004. Springer-Verlag

[9.11] Floerkemeier, C., and Lampe, M. Issues with RFID Usage in Ubicomp Applications. In Pervasive Computing (PERVASIVE) (2004), vol. 3001 of Lecture Notes in Computer Science, pp. 188–193

[9.12] Juels, A., and Pappu, R. Squealing Euros: Privacy Protection in RFID-Enabled Banknotes. In Financial Cryptography (2003), vol. 2742 of Lecture Notes in Computer Science, pp. 103–121

[9.13] Hopper, N. J., and Blum, M. Secure Human Identification Protocols. In Advances in Cryptology - ASIACRYPT (2001), vol. 2248 of Lecture Notes in Computer Science, pp. 52–66.

[9.14] A. Juels and S. Weis. Authenticating Pervasive Devices with Human Protocols. Crypto ’05

[9.15] A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In H.M. Heys and C.M. Adams, editors, Selected Areas in Cryptography, pages 184–199. Springer-Verlag, 1999. LNCS no. 1758

[9.16] Ari Juels. Minimalist cryptography for low-cost RFID tags. In Carlo Blundo and Stelvio Cimato, editors, The Fourth International Conference on Security in Communication Networks - SCN 2004, volume 3352 of Lecture Notes in Computer Science, pages 149--164, Amal, Italia, September 2004. Springer-Verlag.

[9. 17] Lampson, B. W., “Dynamic Protection Structures,” AFIPS Conference Proceedings, 35, 1969, pp. 27–38

[9. 18] Bell, D. E., and L. J. LaPadula, Secure Computer Systems: Mathematical Foundations and Model, Bedford, MA: The Mitre Corporation, 1973

[9.19] DoD, Trusted Computer System Evaluation Criteria (TCSEC), DoD 5200.28-STD.

[9.20] D. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, R. Chandramouli, Proposed NIST Standard for Role-Based Access Control, ACM Transaction on Information and System Security, Vol. 4, No. 3, August 2001, pages 224-274

[9.21] M. J. Covington, M. J. Moyer, and M. Ahamad, "Generalized Role-Based Access Control for Securing Future Applications," 23rd National Information Systems Security Conference, 2000

[9. 22] A. Corradi, R. Montanari, and D. Tibaldi, “Context-based access control management in ubiquitous environments,” Proc. Third IEEE International Symposium on Network Computing and Applications, (NCA’04), pp.253–260, Aug. 2004.

[9.23] G. Zhang and M. Parashar. Context-aware dynamic access control for pervasive computing. In 2004 Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS'04). January 2004.

[9.24] Anand Tripathi, Tanvir Ahmed, Devdatta Kulkarni, Richa Kumar, and Komal Kashiramka. Context-Based Secure Resource Access in Pervasive Computing Environments. In Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW) 2004.

[9.25] Dimmock, N.: How much is `enough'? Risk in trust-based access control. In: IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises - Enterprise Security. (2003) 281-282

[9.26] Cahill, V., et al.: Using trust for secure collaboration in uncertain environments. IEEE Pervasive Computing 2 (2003) 52-61.

[9.27] Adams, W.J.; Davis, N.J., IV. Toward a decentralized trust-based access control system for dynamic collaboration. The IEEE Workshop on Information Assurance and Security, NY, 2005.

[9.28] Roy Campbell, Jalal Al-Muhtadi, Prasad Naldurg, Geetanjali Sampemane, M. Dennis Mickunas, "Towards Security and Privacy for Pervasive Computing." in Theories and Systems, Mext-NSF-JSPS International Sympsoium, ISSS 2002, Tokyo, Japan, November 2002. pp. 1-15, G. Goos, J. Hartmanis, and J. vanLeeuwen (editors) in Lecture Notes in Computer Science.

[9.29] Marc Langeheinrich, Privacy by Design – Principles of Privacy Aware Ubiquitous Systems, in UBICOMP 2001, LNCS 2201, pp 273 291.

[9.30] Alan F. Westin. Privacy and Freedom. Publisher: Bodley Head.

[9.31] Damiani, Vimercati, Paraboschi, Samarati, and Violante, "A reputation-based approach for choosing reliable resources in peer-to-peer networks", 9th ACM CCS 2002

[9.32] Bin Yu, Munindar P. Singh, and Katia Sycara, "Developing trust large-scale peer-to-peer systems", First IEEE Symposium on Multiagent Security and Survivability, 2004

[9.33] Whitby, A., Josang, A. and Indulska, J. "Filtering out unfair ratings in Bayesian reputation systems", AAMAS 2004, New York, USA

[9.34] Weihua Song, Vir V. P hoha, and Xin Xu, "An adaptive recommendation trust model in multiagent system", IEEE/WIC/ACM IAT'04

[9.35] Weihua Song, Vir V. Phoha, "Neural network-based reputation model in a distributed system", pp. 321-324, 2004 IEEE International Conference on E-Commerce Technology (CEC'04), 2004

[9.36] Huang Baohua; Hu Heping; Lu Zhengding, "Identifying local trust value with neural network in p2p environment", The First IEEE and IFIP International Conference in Central Asia on Internet, 2005

[9.37] C. Dellarocas, "The design of reliable trust management systems for electronic trading communities", MIT Working Paper

[9.38] Dellarocas , "Building trust online: the design of robust reputation reporting mechanisms for online trading communities" A combined perspective on the digital era, Doukidis, G., Mylonopoulos, N. and Pouloudi, N. (Eds.), Idea Book Publishing (2004)

[9.39] Dellarocas. "Immunizing online Reputation Reporting systems against unfair ratings and discriminatory behavior", In Proceedings of the ACM Conference on Electronic Commerce, pages 150--157, Minneapolis, Minnesota, USA, 2000

[9.40] Chrysanthos Dellarocas , "Mechanisms for coping with unfair ratings and discriminatory behavior in online reputation reporting systems", In ICIS, pages 520--525, 2000

[9.41] Shigeki YAMADA, Eiji KAMIOKA, “Access Control for Security and Privacy in Ubiquitous Computing Environments”, IEICE TRANS. COMMUN., VOL.E88–B, NO.3 MARCH 2005

[9.42] Lalana Kagal et al, “A Policy Language for a Pervasive Computing Environment”

[9.43] Anand et al., “Enforcing policies in Pervasive Environments”

[9.44] Michael Hecker, Alankar Karol, Christopher Stanton, and Mary-Anne Williams, “Smart Sensor Networks: Communication, Collaboration and Business Decision Making in Distributed Complex Environment ”, proc of International Conference on Mobile Business, 2005, pp. 242-248

[9.45] Mihai Marin-Perianu, Nirvana Meratnia, Maria Lijding and Paul Havinga. “Being Aware in Wireless Sensor Networks”, In 15th IST Mobile & Wireless Communication Summit (IST SUMMIT 2006), June 2006, Myconos, Greece 2006