

# **Self-healing for Autonomic Pervasive Computing**

A thesis submitted to the Department of Mathematics, Statistics and Computer Science and the committee on graduate studies of Marquette University in partial fulfillment of the requirements for the degree of Master of Science

Shameem Ahmed

August 2006

**Marquette University**

**Graduate School**

This is to certify that I have examined this copy of thesis by

Shameem Ahmed

and have found that it is complete and satisfactory in all respects.

---

Dr. Sheikh Iqbal Ahamed

Chair

---

Dr. Douglas Harris

---

Dr. Praveen Madiraju

Approved for the University Committee on Graduate Studies

## **Abstract**

To ensure smooth functioning of numerous handheld devices anywhere anytime, the importance of a self-healing mechanism cannot be overlooked. This is one of the main challenges to growing autonomic pervasive computing. Incorporation of efficient fault detection and recovery in the device itself is the ultimate quest but there is no existing self-healing scheme for devices running in autonomic pervasive computing environments that can be claimed as the ultimate solution. Moreover, the highest degree of transparency, security and privacy should also be maintained. In this thesis, an approach to develop a self-healing service for autonomic pervasive computing is presented. The self-healing service has been developed and integrated into the middleware named MARKS+ (Middleware Adaptability for Resource discovery, Knowledge usability, and Self-healing). The self-healing approach has been evaluated on a test bed of PDAs. An application has been developed by using the service. The evaluation results are also presented in this thesis.

## **Acknowledgments**

First and foremost, I cordially thank my advisor Dr. Sheikh Iqbal Ahamed for his guidance over the last two years. I have benefited tremendously from his insight, his ability to always ask the right questions, and his passion for conducting research that can be used in practical life. He not only served as my supervisor but also encouraged and challenged me throughout my academic program.

My special thanks go to my wife, Moushumi Sharmin, for her love and support in every aspect of my life. I highly appreciate Dr. Douglas Harris and Dr. Praveen Madiraju for their comments and patience. I also want to thank all of my lab members for their valuable suggestions and comments. Finally, my immense gratitude goes to my family for their loving support.

## Table of Contents

<b>LIST OF FIGURES .....</b>	<b>VII</b>
<b>LIST OF TABLES .....</b>	<b>VIII</b>
<b>CHAPTER 1: INTRODUCTION .....</b>	<b>1</b>
<b>CHAPTER 2: BACKGROUND.....</b>	<b>6</b>
2.1. AUTONOMIC COMPUTING .....	6
2.2. CHARACTERISTICS OF AUTONOMIC COMPUTING SYSTEM.....	6
2.3. PERVASIVE COMPUTING .....	7
2.4. AUTONOMIC PERVASIVE COMPUTING .....	7
2.5. SELF-HEALING.....	8
<b>CHAPTER 3: RELATED WORK.....</b>	<b>10</b>
<b>CHAPTER 4: MOTIVATION .....</b>	<b>15</b>
<b>CHAPTER 5: DESIGN OVERVIEW .....</b>	<b>18</b>
5.1. SELF-HEALING SYSTEM OF AUTONOMIC PERVASIVE COMPUTING .....	18
5.2. CLASSIFICATION OF FAULT .....	20
5.3. FAULT DETECTION .....	22
5.4. FAULT NOTIFICATION .....	22
5.5. FAULTY DEVICE ISOLATION .....	22
<b>CHAPTER 6: SELF-HEALING IN AUTONOMIC PERVASIVE COMPUTING .....</b>	<b>24</b>
6.1. FAULT DETECTION .....	25
6.2. FAULT NOTIFICATION .....	28
6.3. FAULTY DEVICE ISOLATION .....	29
6.4. AN ILLUSTRATIVE EXAMPLE .....	31
<b>CHAPTER 7: ATTRIBUTES OF SELF-HEALING.....</b>	<b>34</b>

7.1. CHARACTERISTICS OF SELF-HEALING MODEL .....	34
7.2. ATTRIBUTES OF OUR PROPOSED SELF-HEALING MODEL.....	35
<b>CHAPTER 8: EVALUATION .....</b>	<b>38</b>
8.1. PROTOTYPE IMPLEMENTATION.....	38
8.2. PERFORMANCE MEASUREMENT .....	43
8.3. APPLICATION THAT USES SELF-HEALING MODEL .....	44
<b>CHAPTER 9: CONCLUSION AND FUTURE WORK .....</b>	<b>46</b>
9.1. SUMMARY .....	46
9.2. CONTRIBUTION OF THIS THESIS .....	47
9.3. SHORT AND LONG TERM IMPACT.....	47
9.4. FUTURE WORK.....	48
<b>BIBLIOGRAPHY .....</b>	<b>51</b>
<b>APPENDIX A .....</b>	<b>58</b>
<b>APPENDIX B.....</b>	<b>59</b>

## List of Figures

FIGURE 1.1. SCOPE OF SELF-HEALING IN AUTONOMIC PERVASIVE COMPUTING .....	2
FIGURE 1.2. NETWORKS IN AUTONOMIC PERVASIVE COMPUTING ENVIRONMENT.....	3
FIGURE 4.1. WIRELESS EXAMINATION IN CLASSROOM .....	15
FIGURE 5.1. MARKS+ ARCHITECTURE FOR AUTONOMIC PERVASIVE COMPUTING .....	18
FIGURE 5.2. ARCHITECTURE OF OUR PROPOSED SELF-HEALING SERVICE .....	19
FIGURE 5.3. SCOPE OF SOFTWARE FAULT FOR AUTONOMIC PERVASIVE COMPUTING .....	21
FIGURE 6.1. SELF-HEALING FOR AUTONOMIC PERVASIVE COMPUTING .....	24
FIGURE 6.2. TWO WAY MESSAGE BASED FAULT DETECTION.....	26
FIGURE 6.3. FLOW DIAGRAM OF SELF-HEALING FOR AUTONOMIC PERVASIVE COMPUTING .....	29
FIGURE 6.4. MAPPING OF SERVICE # AND PROVIDER .....	30
FIGURE 6.5. MAPPING OF SERVICE #, SERVICE PROVIDER, AND SERVICE CONSUMER.....	30
FIGURE 8.1. STATUS OF A DEVICE WHEN FAULT DETECTION PROCESS HAS BEEN APPLIED .....	39
FIGURE 8.3. STATUS OF THE BATTERY POWER OF FIVE DEVICES AFTER USING PROPOSED SYSTEM .....	40
FIGURE 8.4. MESSAGE TO THE USER REGARDING LOW PROCESSOR SPEED.....	40
FIGURE 8.5. DEVICE 2 IS RUNNING WITHOUT ANY PROBLEM .....	41
FIGURE 8.6. DEVICE 9 SENDS SOS MESSAGE.....	41
FIGURE 8.7. NO SIGNAL FROM DEVICE 5.....	42
FIGURE 8.8. IMPORTANT FILES NAME SELECTED BY USER TO BE SAVED .....	42
FIGURE 8.9. TIME (MIN) VS. BATTERY POWER (%) FOR MARKS-ORB .....	43
FIGURE 8.10. TIME (SEC) VS. DATA TRANSMISSION (BYTES) FOR MARKS+-ORB .....	44

## List of Tables

TABLE 7.1. COMPARISON OF SELF-HEALING MODELS .....	37
TABLE 8.1. MEMORY FOOTPRINT FOR OUR PROPOSED SYSTEM AND MARKS-ORB .....	44
TABLE 8.2. SCREENSHOT OF WIRELESS EXAM APPLICATION USING OUR PROPOSED SELF-HEALING MODEL .....	45
TABLE DEFINITIONS .....	58

## Chapter 1: Introduction

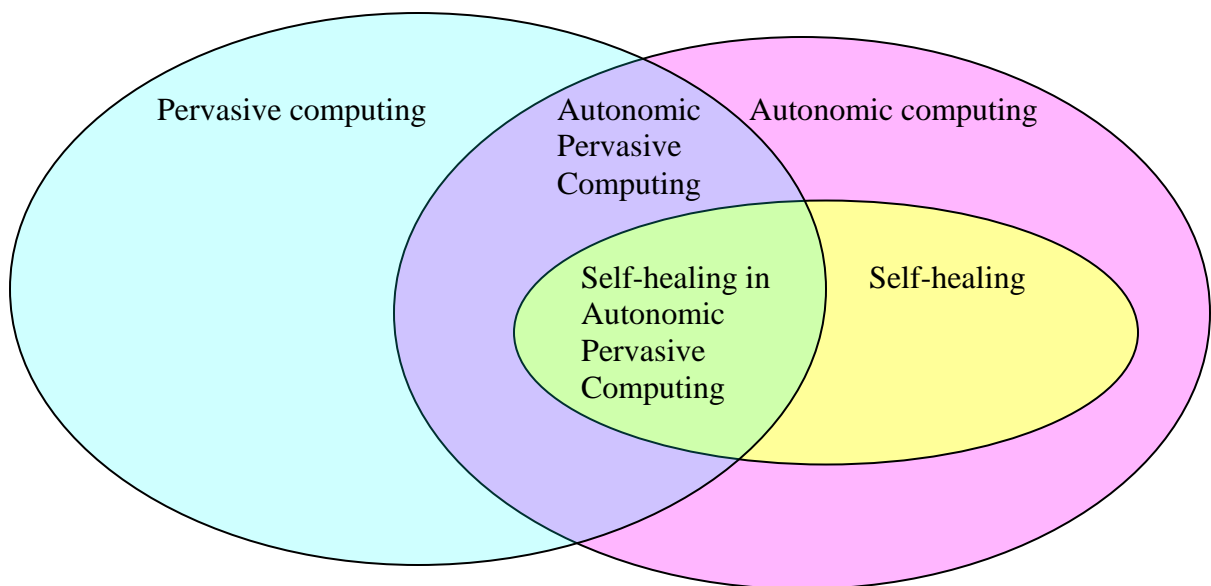
Ubiquitous computing [Weiser93], also known as pervasive computing, has evolved during the last few years due to the rapid developments in portable, low-cost, and lightweight devices. It extends human thought and activity as well as provides a pragmatic world augmented by the behavioral context of its users [Roman02]. Pervasive computing environments focus on integrating computing and communications with the surrounding physical environment for making computing and communication transparent to the users. Pervasive computing includes four major areas mobile computing, wireless networks, embedded computing, and context-aware sensor networks [Robinson04].

Pervasive computing is an emerging field of research. It aims to design and develop models for next generation computing. Remarkable increases in the number of mobile device users, tremendous developments in wireless and mobile technologies, and low cost availability of handheld devices have contributed to the rapid evolution of this computing platform. In this environment things around us are expected to be able to communicate with each other and at the same time have the capability to collect, process, and transport information [Basu05]. Many mobile and handheld devices have become an inseparable part of our life. Pervasive computing has passed its initial stage and the devices running in this environment are now well-known and accessible to everyone. Despite the prevalence of embedded handheld devices, limited processing capability, restricted battery life, inadequate memory space, slow expensive connections, recurrent line disconnection, confined host bandwidth and other problems are the challenges in pervasive computing arena till now [Satya95]. Middleware has evolved to play an important role to cope with these ever-growing requirements.

Systems that have the ability to manage themselves and dynamically adapt to change in accordance with policies and objectives are termed as autonomic computing [Hinnelund04, Kephart05]. This system enables computers to identify and correct problems often before they are

noticed by the user. Autonomic systems have the capability to self-configure, self-optimize, self-protect, as well as self-healing. Systems that have these characteristics are termed as self-managing systems.

Autonomic pervasive computing [Tony05] maintains characteristics from both autonomic computing and pervasive computing environment. Like pervasive computing, devices running in this area should be context and situation aware and these devices form an ad-hoc ephemeral network. These devices are also expected to have the ability to self-optimize, self-protect, self-configure, and self-heal.

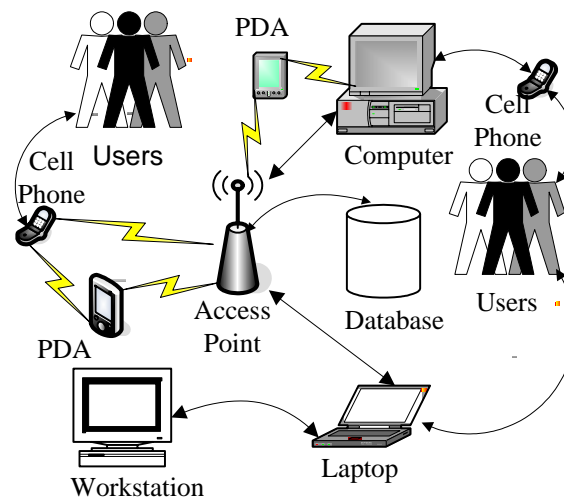


**Figure 1.1. Scope of self-healing in autonomic pervasive computing**

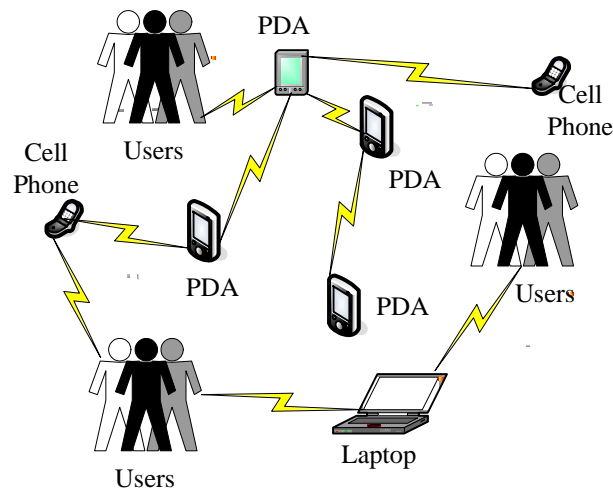
Self-healing describes devices or systems that have the ability to perceive those are not operating correctly and, without human intervention, make the necessary adjustments to restore them to normal operation. A system that continues its operation even in presence of faults is termed as a fault tolerant system [Garlan04]. The concept of self-healing goes beyond fault tolerance since it also provides the device with the capability of recovering from fault by itself or

with the assistance of other devices present in the network. Figure 1.1 depicts the scope of self-healing autonomic pervasive computing.

In an autonomic pervasive computing environment, there are different kinds of networks. There are smart spaces or intelligent environments that provide devices with complex computational support, while there are networks without any such support. In between there are networks that take some kind of support from fixed access points or central servers but are not parts of a smart space.



(a) Ad hoc network in autonomic pervasive computing environment with powerful devices



(b) Ad hoc network in autonomic pervasive computing environment without powerful devices

**Figure 1.2. Networks in autonomic pervasive computing environment**

Figure 1.2 depicts two ad hoc networks in an autonomic pervasive computing environment. In figure 1.2(a), the tiny devices communicate among themselves with the support of fixed, powerful devices. These devices act as servers or proxies and handle complex computations on behalf of the tiny devices. In figure 1.2(b), an ad hoc network is formed by mobile devices. There is no fixed infrastructure support. The devices communicate with each other directly or via another mobile device and are responsible for maintaining computations by themselves. Our research focus is self-healing in infrastructure-less autonomic pervasive computing area.

Considerable research has already been done for fault tolerance in distributed dependable real-time system [Bracewell03]. Some solutions along with prototype for pervasive computing fault tolerant systems have been proposed [Chetan04]. Self-healing autonomous systems are also addressed in [Trumler04]. But no solution has been proposed for a self-healing system in autonomic pervasive computing yet, let alone the implementation of such a system. Since the future technology trend lies in pervasive computing, it is of the utmost importance to have an efficient, transparent, and secure self-healing system. Currently, we are developing middleware named MARKS [Ahmed05, Sharmin05, Ahamed05, Sharmin06], which is suitable for embedded devices running in pervasive computing environments. The Self-healing unit plays a vital role from the above perspectives.

In this thesis, we present an efficient, secure, and transparent self-healing model. Our model is designed for the autonomic pervasive environment, where we assume that the mobile devices would be able to handle necessary computations and communications by themselves without any fixed infrastructure support. We developed its first prototype on a test bed of PDAs, which are connected with short-range ad hoc wireless. Any healing approach will be in vain without proper setup of fault detection and recovery. Efficiency should also not be overlooked. Our self-healing approach is unique from those perspectives too. A modified secret sharing approach [Secret Sharing], not only to cope with the limited storage capacity of the embedded devices but also to

guarantee the security, is being used in our approach. This model provides the third feature (transparency) by performing most of the healing process without users' interference.

The outline of this thesis is as follows: Chapter 2 contains the background information. Chapter 3 contains the current state of the art. Characteristics of our model with motivations behind designing the system are presented in chapter 4. An overview of our proposed approach is illustrated in chapter 5. The details of the models are described in chapter 6. The attributes of our proposed model is presented in chapter 7. The implementation details along with evaluation are provided in chapter 8. We conclude with some novel research directions of future work in chapter 9.

## Chapter 2: Background

This chapter describes all the basics terms of autonomic pervasive computing to make those comprehensible to the readers.

### 2.1. Autonomic computing

According to IBM [Ganek03], autonomic computing will free system administrators from many of today's routine management and operational tasks. Rather than spending valuable time to deal the complexity of computing systems, IBM says that autonomic computing will help corporations to devote more of their IT skills toward fulfilling the needs of their core businesses.

### 2.2. Characteristics of autonomic computing system

Paul Horn [Horn01] illustrates the following characteristics for an autonomic computing system:

- An autonomic system needs to “know itself”.
- In any unpredictable conditions, an autonomic system must be able to configure and reconfigure itself.
- An autonomic system always looks for optimize ways to work.
- An autonomic system must have the capability of self-healing.
- An autonomic system must have the capability for self-protection.
- An autonomic computing system must be able to know its environment and the context surrounding its activity.

According to IBM [Ganek03], to adopt the above characteristics in self-managing system, autonomic computing should have the following four attributes:

**Self-configuring:** Systems adapt automatically to dynamically changing environments.

**Self-healing:** Systems discover, diagnose, and react to disruptions.

**Self-optimizing:** Systems monitor and tune resources automatically.

**Self-protecting:** Systems anticipate, detect, identify, and protect themselves from attacks from anywhere.

### 2.3. Pervasive computing

Pervasive computing is computation that's freely available everywhere [Weiser93]. Pervasive computing environments focus on integrating computing and communications with the surrounding physical environment to make computing and communication transparent to its users. The pervasive computing environment is comprised of various handheld devices that include but not limited to PDAs, cell phones, smart phones, laptops, etc.

All the applications running in a pervasive computing environment have the following characteristics [Abowd00, Yau02]:

- **Context-sensitivity:** Ability of a device to sense its present context and changes in contextual data.
- **Situation-awareness:** Capability not only to capture but also to analyze the relationship among various contexts and actions over a specific of time span.
- **Ad-hoc communication:** Instant establishment and termination of communication channels among application with the change of contexts. Ad-hoc networks do not use built-in network infrastructures and centralized administration [Broch98].

### 2.4. Autonomic pervasive computing

Autonomic pervasive computing is a relatively new term. It is discussed by Tony *et al.* in [Tony05] and they defined autonomic pervasive computing as follows:

Autonomic Pervasive Computing is pervasive computing systems whose context-aware adaptivity is subject to human governance. It spans both autonomic computing and autonomic

communications (managing highly adaptive networks such as sensor networks and mobile ad hoc networks). They differentiate “autonomic pervasive computing” from “autonomic computing” using the “adaptivity” point of view. Autonomic pervasive computing is highly ad hoc and heterogeneous, both technologically and organizationally.

We classify autonomic pervasive computing into three categories:

1. Fixed-infrastructure based
2. Infrastructure-less (pure ad-hoc)
3. Hybrid

In fixed-infrastructure based autonomic pervasive computing environment, devices get support from powerful servers, access points, etc. These powerful devices handle complex computation and communication on behalf of the mobile devices. Smart/active spaces are examples of this type of environment. In an infrastructure-less autonomic pervasive computing environment, mobile devices are responsible for communication and computations. They communicate with each other using a short-range wireless network. An ad-hoc network in a stadium is example of this type of network. There are also hybrid networks that have some support from the infrastructure. Ad-hoc networks with the ability to communicate with the access point fall into this category. In this thesis, we focus on infrastructure less autonomic pervasive computing environment.

## **2.5. Self-healing**

Self-healing systems are able to recover from faults by themselves or with the assistance of other devices of the same network. Fault tolerant systems are systems that can continue to operate even in the presence of faults [Garlan04]. All self-healing systems are fault tolerant systems as they can detect faults, notify other devices of these faults, and try to recover from these faults. In an autonomic pervasive computing environment, self-healing is a hard problem as most of the

devices that operate in this area are resource-poor. The lack of fixed infrastructure and the ad-hoc nature of the network make the self-healing problem more challenging.

## Chapter 3: Related Work

Self-healing is an essential component of every computing system. It is an integral part of devices running in an autonomous pervasive environment, as the main focus of this area is to make users free of operating details. It is a widely researched topic in the field of distributed computing, grid computing, and autonomous computing. In each of these areas, many schemes are proposed that attack this problem from various standpoints. Researchers are working on several policies like architecture-based system [Garlan02, Dashofy02], infrastructure based approach [Appavoo02] for long. Most of these proposed models are suitable for physically connected computers in distributed computing environment. There is no established method that provides a solution for devices running in an autonomous pervasive computing environment where it is assumed that the devices are connected to each other wirelessly. The autonomous computing field demands that devices should be able to self-configure, have self-healing, self-optimization, and self-protection capability. On the other hand, the main attributes of pervasive computing are context-awareness, situation-awareness, and ad-hoc networking. The focus of this thesis is self-healing for autonomous pervasive computing environment that contains all the above-mentioned characteristics.

Soila and Priya [Pertet04] presented proactive recovery in Distributed CORBA applications. They did not concentrate on fault-prediction technique; rather they focused on the exploitation of fault prediction in systems that had real time deadlines. Our system deals with pervasive computing and it can predict a fault by calculating different states of the system.

To handle transient software failures, a proactive approach named software rejuvenation, was proposed by Huang *et al.* [Huang95]. According to this approach, if errors are accumulated beyond a threshold, then it will kill and re-launch the application. A lot of works about rejuvenation policies, to increase system availability and to reduce the cost of rejuvenation, were done by [Bobbio98, Garg98, Yujuan03]. However, to hand-off the existing state of the faulty

device just after its re-launching was overlooked here. Our approach preserves this state among other devices in a secured manner so that the healing manager can help the faulty device to get its actual state after healing.

Garlan and Schmerl presented a system [Garlan02] that uses an architectural model for monitoring, problem detection, and repair. In their model an executing system is monitored to observe its run time behavior. Monitored values are compared with the properties of an architectural model. The architectural model triggers constraint evaluation if the system is operating outside an envelope of acceptable ranges. These violations of constraints are handled by a repair mechanism according to the architecture and these changes are propagated to the running system. This architectural model is represented as a graph of interacting components where nodes are termed as components. These nodes represent the principal computational elements and data stores of the system: clients, servers, databases, user interfaces, etc. These nodes are interconnected by connectors, which are complex bases of middleware and distributed systems. These powerful device requirements make this model unusable in infrastructure-less pervasive computing environment.

Eric *et al.* [Dashofy02] describes a system based on software architecture that uses software components and connectors for repair. To dynamically repair a system they concentrate on the current architecture of the system; to express an arbitrary change to that architecture; to analyze the result of the repair; and to execute the repair plan on a running system without restarting the system. To follow this approach they need complete information about the devices and software running in those devices. In a autonomic pervasive computing environment this type of information is not available. Their system is targeted for connected distributed environment where system level information is available. They also use infrastructure support for repair purposes.

Gordon *et al.* [Blair02] in their paper presented an analysis of the role of “Reflection” to support self-healing systems. They also suggested that middleware would be the appropriate

place for including self-healing unit. They offered their primary analysis based on distributed systems. But they did not implement the middleware and also not the self-adaptive, self-healing unit.

AMUN (Autonomic Middleware for Ubiquitous Environment) [Trumler04] is a middleware that deals with self-healing for ubiquitous environment. The AMUN self-healing and self-organizing unit consists of four main parts: the transport interface, the event dispatcher, the service interface and service proxy, and the autonomic manager. The autonomic manager is the principal unit for managing communication between other units. But the main concentration of this project is an indoor environment like inside an office building. They use “Smart Doorplates” that use and display situational information of the owner of the office. This idea restricts its use only in a smart environment.

L. Kant [Kant02] proposed a self-healing mechanism for wireless networks. He claimed that this mechanism could provide seamless restoration of affected services due to random/sporadic network facility failures. Also this model considers a fixed telecommunications system only and concentrates on restoration of the faulty system in network layer. This approach is not suitable for pervasive computing since the ad-hoc nature is totally overlooked here.

Y. Tohma in [Tohma02] described the challenges and proposed solutions to achieve fault tolerance in autonomic computing environment. This solution creates groups of similar devices and the neighboring devices decide whether a particular device is faulty or not. He also proposed to keep three copies of information to recover fault. This proposed method is expensive and not applicable in an ad-hoc environment as in this environment the group formation is difficult and neighbors are changing very frequently.

Mills *et al.* in [Mills04] presented an algorithm for fault detection for autonomic systems. A two-way heartbeat failure-detection technique is used. A monitor is used to receive periodic messages from a number of monitorables. The time length to receive a message is used as the

detection criteria for failure. This system is simple and is designed for connected distributed systems where the devices will not change. Device mobility issues are also ignored here.

Brown and Redlin in [Brown05] proposed a benchmarking system for measuring the effectiveness of self-healing mechanism in autonomic systems. They concentrated on powerful systems that are rarely present in pervasive ad-hoc networks.

In [Poladian06], Poladian *et al.* proposed a task-based adaptation technique for an ubiquitous computing environment. This system supports heterogeneity, resource variability, mobility, ubiquity, and task-specific user requirements. This self-adaptation infrastructure has three distinctive features that allow explicit representation of user tasks and provides an environment management capability to translate user tasks and also provides a formal basis for understanding the resource allocation, and support optimal allocation at run time. Though this system is targeted for pervasive environments, it needs infrastructure support for fulfilling all the above activities which makes it unusable for truly mobile pervasive environment.

In [Chetan04], Chetan *et al.* have classified faults, pointed out various research challenges, and have also proposed solutions to some of the challenges in a pervasive environment. Their fault tolerant system uses context information to tolerate application and device faults. They considered a fail-stop fault model consisting of device and application faults. In this approach, if an application fails, it is restarted. The fault model only considers application failures caused by device failures, network faults, and failures due to faulty usage. This system is designed for active spaces.

Most of the work in the self-healing area is done either for distributed systems or for active/smart spaces. The area of autonomic pervasive computing is void of powerful device support. The devices running in this environment are resource poor and these are expected to handle the necessary computations and communications with limited battery and processing power. These features make self-healing in autonomic pervasive computing a hard and unique

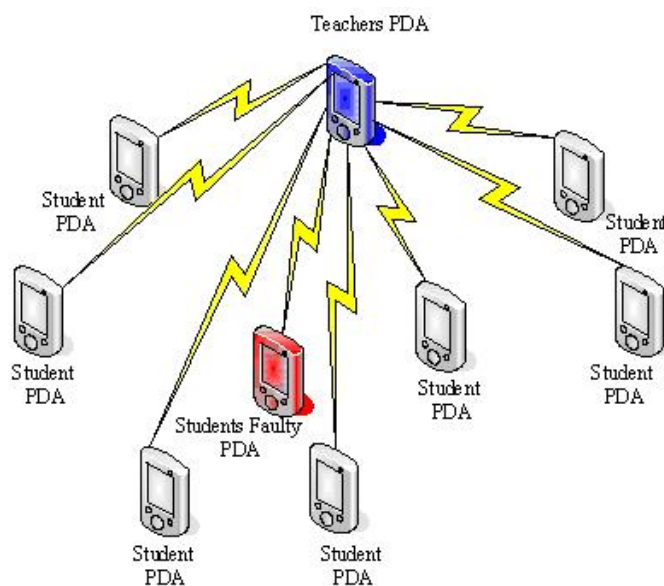
problem to solve. To the best of our knowledge, there is no work till now in the area of self-healing for autonomic pervasive computing.

## Chapter 4: Motivation

The autonomic pervasive computing area is now strong and powerful as the use of handheld and wearable devices is increasing in a rapid rate. These devices are capable of communicating with other devices and run various applications like powerful devices. People are using these tiny mobile devices all the time and everywhere. If these devices operate incorrectly or prompt users for each little malfunctioning, then the usability of these devices will reduce. Hence, self-healing comes into play as self-healing helps to execute a system uninterruptedly. Self-healing is an integral part of autonomic computing systems. Here, we consider some scenarios to show the importance and applicability of self-healing in autonomic pervasive computing environment.

### *Scenario 1*

A group of high school students appear in a wireless examination. After getting the questions in their PDAs from their teacher Dr. John's PDA (let X, the healing manager), they start their tests. During the exam, all on a sudden, one student's PDA (let Y) starts unusual behavior. Figure 4.1 presents the exam scenario.



**Figure 4.1. Wireless examination in classroom**

## *Scenario 2*

Returning from a visit of a museum, a group of high school kids want to share their experience (stored in their PDAs) to enrich their knowledge. One device of the network is having a high probability of going down and the device owner wants to store some of its important information for future use in other devices.

The above scenarios present situations where healing is needed. But these scenarios raise the question that do we need to inform the user each time a device is having problem? Can we fix some of these problems transparently and securely? Are we sacrificing security and efficiency issues?

The first scenario can occur in any classroom. This problem can be solved by calculating the rate of changes of all of the status of the faulty device. If the device (Y) finds out that the fault is due to the malfunction of a running application, then without any delay, it can send an SOS message along with the answer files. The teachers' device (X) can isolate Y from the entire network by removing all entries of Y as a service provider. By this time, Y can inform the device owner about the problem. By using the system interrupt, it can kill the problem causing application.

In the second scenario, to avoid the loss of data stored in that device, the healing manager can disseminate the stored information to the remaining devices in a secure manner. By consulting the logbook, necessary measures can be taken to restore the device's prior working state. The disseminated information content can be used later to restore the device to help it to work to its full extent.

A self-healing model can solve the issues of the above situations. To cope with the challenges presented by the pervasive ad-hoc network, we feel that a self-healing model should be lightweight, energy-efficient, and infrastructure less. In this thesis, we present a self-healing model, which is efficient, secure, and works transparently. The required characteristics of a self-

healing model are: No infrastructure support (powerful servers, proxies, etc.), lightweight in terms of executable file size, Non-degradable performance, Energy efficient, Transparent, Secure.

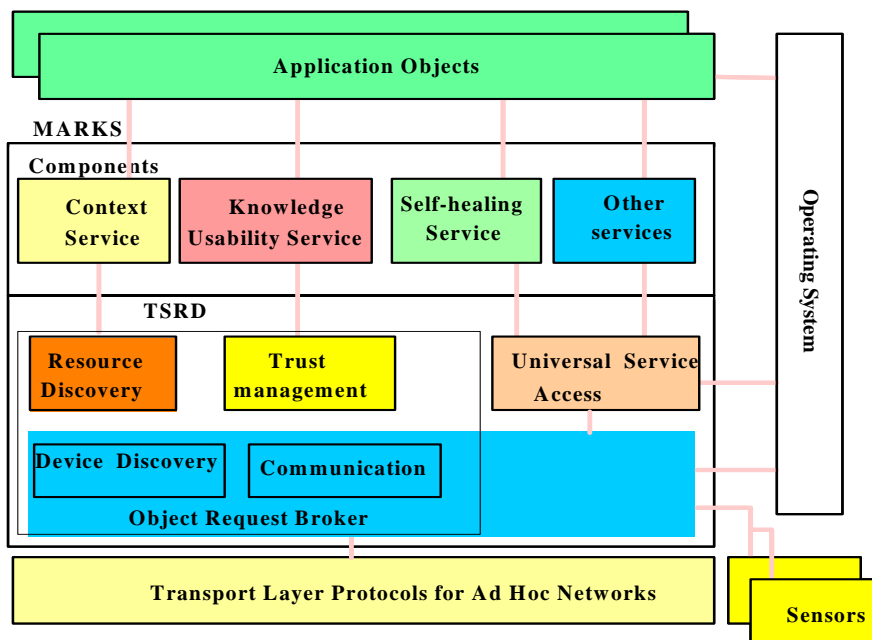
The details of the above characteristics are discussed in Chapter 7.

## Chapter 5: Design Overview

We describe the main properties of a self-healing system of autonomic pervasive computing in section 5.1. The classification of faults is described in section 5.2. Brief discussions on fault detection and fault notification are presented in section 5.3 and 5.4 respectively. How faulty devices should be isolated from the existing network is depicted in section 5.5.

### 5.1. Self-healing system of autonomic pervasive computing

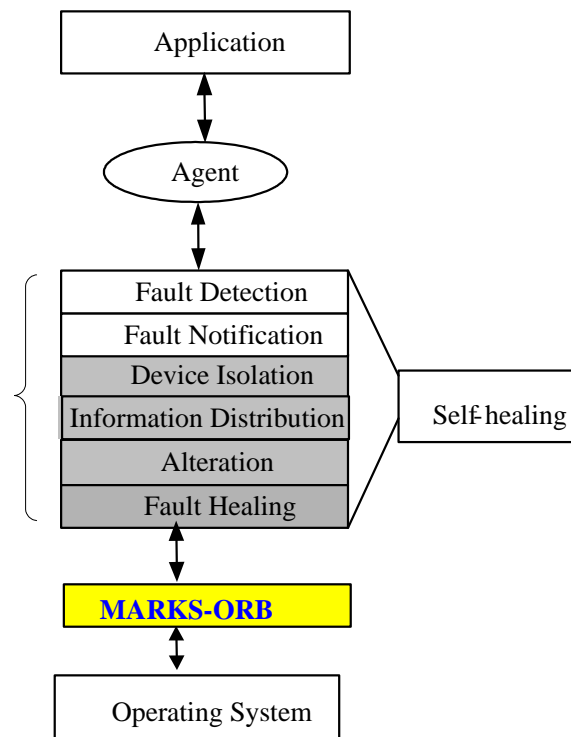
Our self-healing service is an integral part of MARKS+, which is the extended version of MARKS [Sharmin06] (Middleware Adaptability for Resource Discovery, Knowledge Usability, and Self Healing) which is our developed middleware that incorporates different kind of services. The fault detection, fault notification, and faulty device isolation are taken care of by the self-healing service. Figure 5.1 shows the MARKS+ architecture along with the self-healing service.



**Figure 5.1. MARKS+ architecture for autonomic pervasive computing**

An effective self-healing service should address the following challenges:

- No regular functionality of the network will be hampered due to any fault of any device.
- All significant information of the faulty device should be preserved in secure fashion.
- The device will be facilitated to heal its fault by itself or at best with the assistance of other devices of that network.
- After reviving, the faulty device should be able to regain its previous states in such a way that it should feel there was no fault.



**Figure 5.2. Architecture of our proposed self-healing service**

To address the above challenges in an apposite manner, our proposed self-healing pursues quite a few steps:

1. Fault detection
2. Fault notification
3. Faulty device isolation
4. Alteration

## 5. Information distribution

## 6. Fault healing

Figure 5.2 shows the architecture of our proposed self-healing service. The fault-detection and fault-notification unit is used by each device running MARKS+. The fault detection unit may be utilized by both the healing manager and the device itself. But the other three units (faulty device isolation, alteration, information distribution) should be handled by the healing manager. In this thesis, we only concentrate on three steps: fault detection, fault notification, and faulty device isolation.

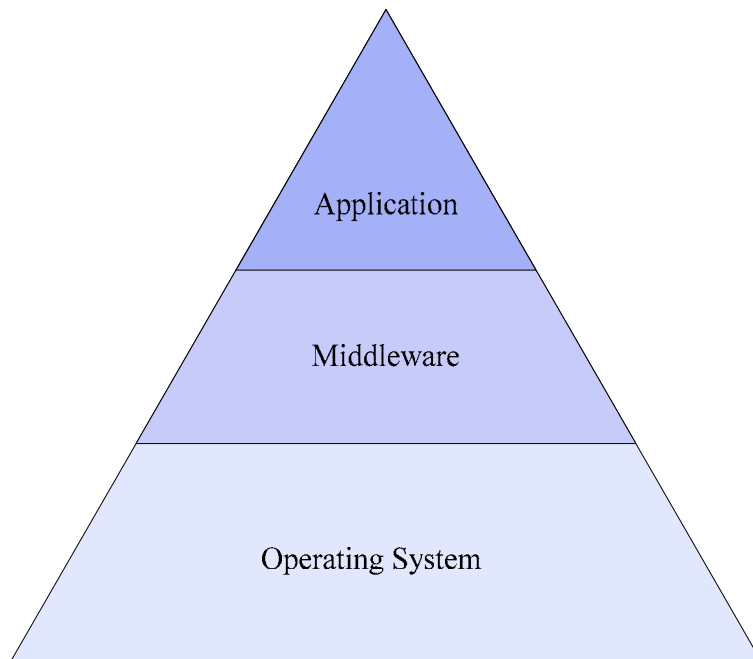
### 5.2. Classification of Fault

Fault can be classified from different perspectives:

- **Hardware fault:** This fault is mainly caused by physical constraints of the devices running in autonomic pervasive computing environment. In our research, we include the following faults as hardware faults:
  - Low battery power: Battery is the main power source for mobile devices. This fault occurs due to insufficient battery power.
  - Limited signal strength: Wireless communication is one of the main communication medium for mobile devices. While the signal strength is low, in most cases, the mobile devices are not able to communicate with each other and the entire network starts malfunctioning.
  - Insufficient disk space: Sometimes mobile devices fail to work properly due to insufficient disk space. We are considering this fault as a hardware fault.
  - Byzantine fault [Schneider84, Byzantine Problem]: These faults encompass those faults that are commonly known as "crash failures" and "send and omission failures."
 

A system may respond in an unpredictable way, if there is any Byzantine fault.
- **Software Fault:** It includes the following faults:

- Application fault: Application fault occurs due to problem in any kind of software application that is running in the mobile devices in autonomic pervasive computing environment.
- Middleware Fault: Middleware is the software layer between the operating system and the applications [Middleware]. It plays a very important role in autonomic pervasive computing environment. Any fault that occurs due to problem in middleware is called middleware fault.
- OS fault: There are some faults that occur in OS level. These faults are named as OS fault.



**Figure 5.3. Scope of software fault for autonomic pervasive computing**

Software fault follows the pyramid approach, which is described in figure 5.3.

- **Prioritized Fault:** According to priority, we have classified faults into the following categories:
  - Low priority fault

- Medium priority fault
- High priority fault

The healing manager has a queue for different kind of faults reported by devices. The priority has been assigned to help the healing manager. High priority faults should be handled first, then medium priority faults, and finally low priority faults.

- **Communication Fault**: All types of fault related to wireless communication belong to this category. Network failure is one kind of communication fault.

### **5.3. Fault detection**

High-quality fault detection, the first step of a self-healing process, not only prevents loss of resources but also lessens healing time. To ensure supreme-quality, our proposed Self-healing approach periodically monitors as well as assembles the status of all of the running applications, memory, power, communication signal etc. Drastic changes in those values will generate faults. By using the rate of change of these over time, it tries to figure out the existence as well as the cause of a fault, if there is any.

### **5.4. Fault notification**

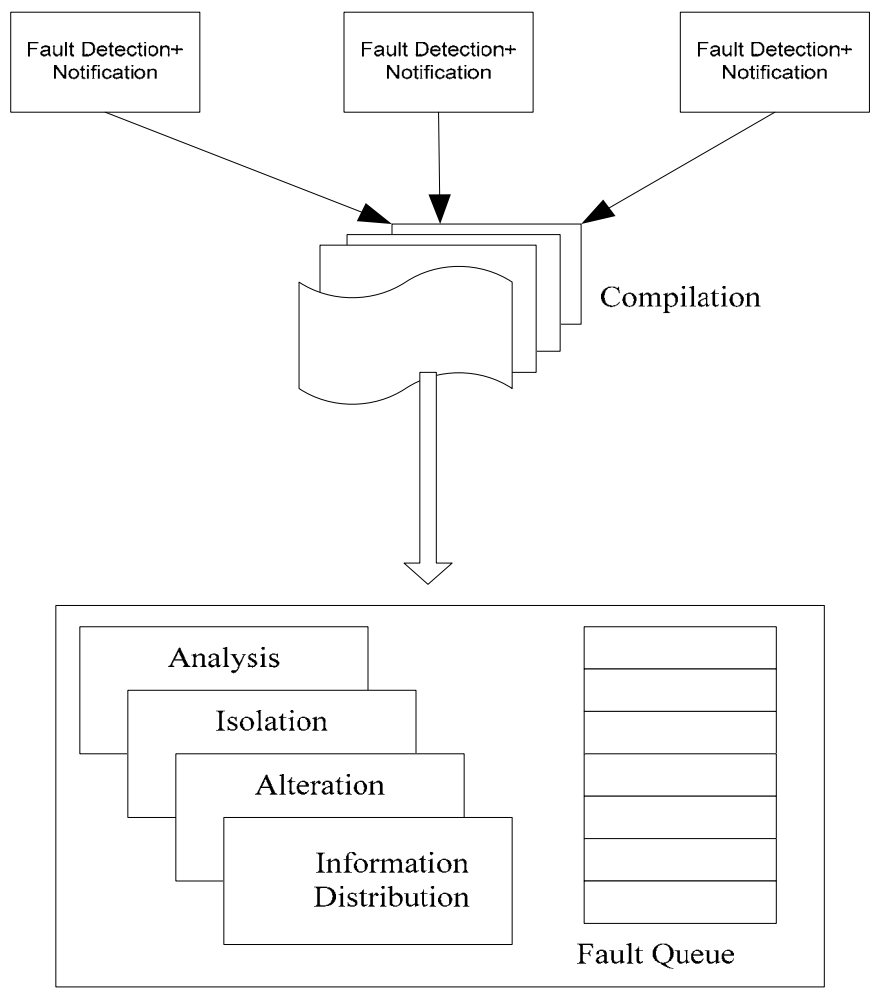
How can a device inform the healing manager that it is in fault? If the device is in fault then is it really possible to inform of the fault? In a distributed computing system, heart beat message and absence of heartbeat message are used to notify of the fault. In autonomic pervasive computing, we have applied the similar concept but in a modified way. We have introduced different kinds of message systems like OK message, SOS message etc. to inform the healing manager not only of the presence of a fault but also the cause of the fault.

### **5.5. Faulty device isolation**

The isolation of faulty devices from the remaining network, a grand challenge of fault tolerant as well as self-healing system, can be achieved in self-healing in a very simple way. In case of MARKS+, every device is mapped with another one by means of service availability in these devices. It is adequate to remove the entry of the faulty device from that mapping to ensure its isolation from the entire network. The detailed description of our approach is present in the following chapter.

### Chapter 6: Self-healing in autonomic pervasive computing

Self-healing is the process of detecting faults, notifying those, and also to recover from the faults without human intervention. This system makes necessary adjustments to restore the devices' normal operating condition. Figure 6.1 portrays the high level view of a self-healing system.



**Figure 6.1. Self-healing for autonomic pervasive computing**

According to figure 6.1, all the faults reported from various devices will be compiled first and then will be stored in the fault queue of the healing manager. According to the fault priority, the

healing manager decides which fault should be handled first. Then the healing manager will do the analysis of the fault, isolate the faulty device from the existing network, and will also do the service alteration and information distribution. In our thesis, we mainly concentrate on three steps of self-healing: fault detection, fault notification, and faulty device isolation.

The fault detection unit is responsible for detecting any kind of fault of a faulty device. This unit allows calculation of different physical properties (memory, power, signal strength, etc.) of the device. How to notify the fault to the healing manager is handled by the fault notification unit. Section 6.1 describes the details of the approach for fault detection of a faulty device. Fault notification has been described in section 6.2. The details of the faulty device isolation unit have been included in section 6.3.

## 6.1. Fault detection

Fault detection is the first step of any self-healing system. Detecting any kind of fault in a device is normally known as fault detection. Here is the formal definition of fault detection:

*Device Status:* Let  $Z_t(x)$  be the status of a device at time  $t$ , where  $x$  represents an arbitrary input vector [e.g. rate of change ( $dy/dt$ ) of power, memory, communicational signal etc. over time]

*Test:*  $T = \{v_1, v_2, \dots, v_n\}$  where  $v_1, v_2, \dots, v_n$  are input vectors and  $Z_t(v_i)$  represents the status of the device.

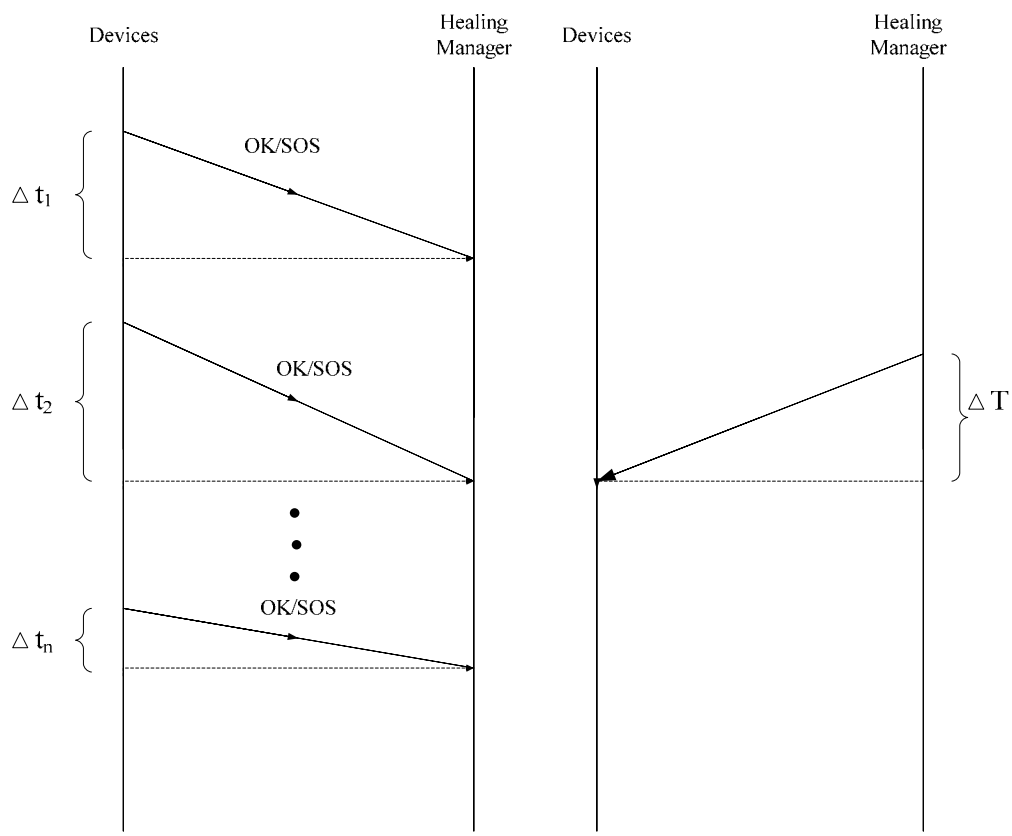
*Fault Detection:*  $T$  detects a fault in the device iff  $[(Z_t(v_i) \sim Z_{t+1}(v_i)) > \text{predefined threshold value}]$

For example, if the change of signal strength is 10% at time  $t$  and change of memory space is 25% at time  $(t+1)$  and if the threshold value is 12% then as the rate of change of memory space is greater than the threshold value, according to our approach, there should be a fault.

### 6.1.1. Two way message based fault detection

According to figure 6.2, each device (except the healing manager) will send OK/SOS message after a specific time period. This period is called the heartbeat period (Hp). For each

device, it takes  $\Delta t_i$  time, where  $i$  is the device number. So, the healing manager will get the OK/SOS message after  $(H_p + \Delta t_i)$  time. Not only that, the healing manager itself will periodically ( $\Delta T$ ) broadcast another message to inform all the existing devices about its aliveness. If the healing manager doesn't get any message from any device within a specific time period then the healing manager will assume that the device is faulty and it might need some help. On the contrary, if the healing manager doesn't broadcast its message, all the devices along with the service manager will understand that the healing manager itself is in fault and the service manager will take the responsibility of the healing manager.



**Figure 6.2. Two way message based fault detection**

K. Mills *et al.* [Mills04] described two-way heart beat based failure detection technique for distributed system. Our environment is different than pure distributed system. We concentrate only on the autonomic pervasive computing environment.

### 6.1.2 Network bandwidth consumption

For a small network in autonomic pervasive computing environment, there won't be so many devices and hence the bandwidth consumption might not be prominent for the network itself. However, if we consider a large network, we should have such a system which would be able to utilize the full network bandwidth. If the message size to send to healing manager is  $M_D$  and the message size of the healing manager to device is  $M_H$  then the system consumes bandwidth,  $B$  which is

$$B = (N * M_D + M_H) / Hp$$

To keep this consumption stable, the healing manager must process  $N/Hp$  messages. Since the network is ad-hoc in nature and a device can join or leave at any time, the system should adopt the size of the heartbeat period ( $Hp$ ). The algorithm to adjust  $Hp$  is presented below.

#### **procedure AdjustHeartbeat (N, Nmax, C, Hmin, Hmax)**

$N$  = Total number of devices (except the healing manager)

$Nmax$  = Maximum allowable number of devices

$C$  = Bandwidth capacity

$Hp$  = Heartbeat period

$Hmin$  = Predefined minimum value of heartbeat message

$Hmax$  = Predefined maximum value of heartbeat message

L1. **if** a new device joins **then**  $N++$ ;

L2. **if** an existing device leaves **then**  $N--$ ;

L3. **if** ( $N > N_{max}$ ) **then**

$N--;$

**return**

L4. **if** ( $N < 1$ ) **then**

$N++;$

**return**

L5.  $H_p = N / C$

L6. Broadcast  $H_p$  to all the existing devices

**end procedure**

## 6.2. Fault notification

Not only to push any information but also to notify its aliveness or its fault, the device itself needs to communicate with the healing manager periodically. In Gaia, [Chetan04] a heart beat message mechanism is used only to indicate the aliveness of any device. The absence of a heart beat message implies the existence of a fault. Thinking ahead a little bit more, we have incorporated a generic message passing scheme not only to facilitate the function of a heart beat message but also the efficacy of an *SOS message* for helping the healing manager to be informed about the faulty device's current situation. In this scheme, each device will send any one of the following messages to the healing manager:

- *OK message*: It simply sends a packet containing "OK" string. It's nothing but a heart beat message.
- *SOS (Save Our Soul) message*: After identifying any fault in its own device, the self-healing unit of that device sends *SOS message* which may include some file names along with that message. An example of such type of message is: "SOS, exam3cosc060, log

status”. This means that the faulty device is requesting to save files named “exam3cosc060” and “logstatus”.

- If the healing manager doesn’t get such a message for a pre-fixed threshold period of time, right away it will commence the next steps (device isolation, information distribution, alteration) assuming that the device is in fault. If the healing manager gets *SOS message* along with some file names, then healing manager will initiate to get the files from the device and will store those among other devices in a secured distributed manner.

Figure 6.3 shows the flow diagram of fault detection and fault notification.

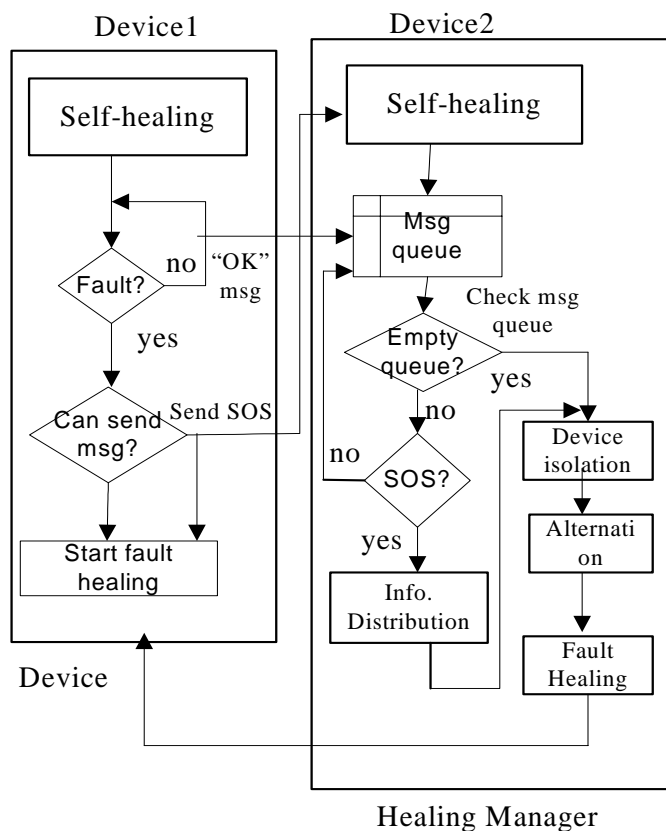


Figure 6.3. Flow diagram of self-healing for autonomic pervasive computing

### 6.3. Faulty device isolation

We have deployed a very simple approach to isolate a faulty device from the remaining network. In our approach, each device is mapped with another device based on service availability. So if we can remove the entry number from the mapping list, the faulty device can be isolated from the entire network.

Figure 6.4 shows the mapping of the service # and the service provider. We have followed a standard for the service #. For example, service # 1 means “internet service”, service # 2 means “office software”, service # 100 means “music software”, etc. Self-healing approach incorporates a list named serviceList by which the service can easily be identified. For example, serviceList (100) will return “music software”. Figure 6.5 exemplifies the three-dimensional mapping of service provider, service consumer, and service #. Here, D1 means Device 1, D2 means Device 2, etc. These mappings are implemented in our proposed self-healing by using a hash table. By means of a standard “remove” function, our self-healing system can remove the entry of a faulty device from the hash table as well as from the entire network.

Service #	Service provider			
1	D1	D7	D9	D3
2	D19			
3	D12	D12	D2	
4				
5	D1	D3	D4	D9
.....	D20			
100	D2	D9	D18	

**Figure 6.4. Mapping of service # and provider**

Service provider	Service consumer						
	D1	D2	D3	...	D9	..	D20
D1		1			1, 5		
D2			10 0		100		
D3	5						
D9		1,5					5
..							
D20	15		99				

**Figure 6.5. Mapping of service #, service provider, and service consumer**

#### **6.4. An illustrative example**

To make our proposed self-healing system more comprehensible, here we are giving an illustrative example. In a wireless exam system, a teacher will conduct an exam. The students will give the answers and will send the answers to the teacher's PDA. The requirements for teacher and students are as follows:

##### ***Requirements from Instructor point of view***

- 1) The system should enable the instructors to enter a certain number of questions. We assume that the instructor will be teaching other courses as well and hence may plan to create questions for the other courses concurrently. Hence, one set of questions should not overlap with others. To ensure this, all the questions for a particular course should be grouped together and stored in a data structure or database (whatever is efficient on a PDA).
- 2) The system should facilitate the creation of exams. It can have any number of
  - a. Multiple choice
  - b. True/false
  - c. Short answer (fill-in) and
  - d. Open ended response options.
- 3) The instructor should be able to create a suitable "answer key" for the questions. Ideally, an instructor will view the question and type the answer so that the answer is visible whenever the question is displayed. Teachers may want the correct answer to be sent to the student after grading is complete.
- 4) The system should have the capability to do the auto grading (multiple choices, fill blank, and true-false) and generate result (text, table and graphical output) of varying format.

- 5) The distribution of the entire exam should take place when the instructor selects the designated action.
- 6) The system should also provide the facility to distribute the grades to the students.
- 7) The instructor would be the only authorized person to specify the time limit for the students to submit their answers. The students can commit their answers to the instructor before the expiration of the given time frame. Otherwise, the system should have necessary mechanisms to deny access to the questions after the pre-defined time is over, and commit all the answers to the instructors PDA identifying the students who answered questions.
- 8) The instructors PDA must be able to store the data received in an efficient way. The storage constraint imposes the prime challenge in this issue.
- 9) The instructor will collect feedback from the students during and after any exam. The instructor can exercise the option of filtering stored feedback for repeated use.
- 10) This system can be easily transformed into a survey instrument. The requirements for designing, administering and collecting survey instruments are easily identifiable by the instructor.

***Requirements from Student point of view***

- 1) The system should facilitate answering of questions. They should be able to submit their answers within a pre-specified time limit.
- 2) Normally students get tensed up during any exam. It is very reasonable from their perspective to expect the PDAs to be reliable. The devices should not play any tantrums (like denying the students access to network, resetting in the middle of an exam, partial download of file because of unspecified reasons). An occurrence of a single event specified above will undermine the students and instructors confidence in the entire process. All these considerations obviously have to be taken care of.

3) The students should also be able to submit their comments during and after the exam. While they would not be able to submit the answer outside the time limit imposed by the instructor, they can comment at any time at any topic related to the course and question.

#### **6.4.1. Fault detection in wireless exam**

Each student's device is running our self-healing system. If a student's device becomes faulty (e.g. hardware fault, software fault, communication fault, etc.), it will be detected through our proposed system.

#### **6.4.2. Fault notification in wireless exam**

As soon as the student's device finds the problem, it will send OK or SOS (Save Our Soul) message to the healing manager (here teacher's PDA). The healing manager will take necessary steps according to the message it gets from the faulty device.

#### **6.4.3. Faulty device isolation in wireless exam**

After identifying the faulty device, the healing manager will remove that faulty device from the current network temporarily so that the other devices of that network can work perfectly.

## Chapter 7: Attributes of Self-healing

Self-healing in an autonomic pervasive computing environment has some unique research challenges. In this chapter we discuss the required characteristics of a self-healing model appropriate for this environment. We also present how our proposed scheme complies with these characteristics.

### 7.1. Characteristics of self-healing model

A self-healing model targeted for autonomic pervasive computing should have the following attributes:

- 1) ***Infrastructure less.*** No infrastructure support (powerful servers, proxies, etc.) should be required. If the focus is on truly pervasive environments then the model should work independently without any external support as in this environment infrastructure support is not always available.
- 2) ***Lightweight.*** The model should be lightweight in terms of executable file size.
- 3) ***Non-degradable performance.*** The model should not put much overhead on the performance of the device.
- 4) ***Energy efficient.*** Self-healing models should be energy efficient. It should not require much battery power for computation or communication purposes.
- 5) ***Transparent.*** The main idea behind designing autonomic systems is transparency. Every self-healing model should have some level of transparency. However, it should also inform the user about critical system information.
- 6) ***Secure.*** Self-healing systems require information distribution and backups to recover from faults. The information distribution and storage should be highly secure to maintain user privacy and security.

## **7.2. Attributes of our proposed Self-healing model**

In this section we discuss the attributes of our proposed model and also talk about how these attributes fulfill the requirements of autonomic pervasive computing environment.

### **7.2.1. Efficiency**

Our system is efficient from both memory and speed points of view. One replica of the healing manager is preserved in service manager. It does not consume much memory in this approach since it does not need to keep a replica of all the devices, while this is a common scenario for most of other schemes. Moreover, MARKS-ORB and the fault detection and notification system occupy a very little space in the PDA.

According to the flow diagram of a self-healing service, there is a loop for message queue inside of which the fault detection and notification system lies. The overall time complexity of this system is  $O(n)$ . On the contrary, Device discovery and Communication, the two functions of MARKS-ORB are independent of each other and that is why the time complexity for MARKS-ORB is  $O(n)$ . This low time complexity makes our approach efficient from the speed perspective.

### **7.2.2. Transparent**

Fault detection and notification tries to assure the nominal involvement of the user. In most cases, without any kind of user interruption, the healing manager can detect a fault. It involves the users only when decision largely depends upon the users' preference.

### **7.2.3. Infrastructure less**

Our system doesn't require any fixed infrastructure to run. All the algorithms are running on different mobile devices (e.g. PDA, cell phone, etc.) using wireless communication technology.

#### **7.2.4. No degradable performance**

Our system doesn't create any adverse affect on the devices. We have proved those from the perspective of different performance measurement metrics like battery power, memory space, and signal strength. The details have been described in the next chapter (Evaluation).

#### **7.2.5. Secure**

Currently, we didn't incorporate any security features in our system. However in near future, while we develop the full fledged self-healing system, we are planning to deploy security systems in different stages.

##### ***Authentication: In Healing Manager and other devices***

To provide a high degree of security, a simple yet effective secret code system will be devised. No one will be able to use the device without knowing the secret code.

##### ***Security: Regarding Information Distribution***

A modified secret sharing approach, can be implemented in a very simple fashion. A random number procedure might be chosen to generate the key that is XOR-ed with all the information that should be distributed among N devices. Only  $t$  ( $t \leq N$ ) devices are needed to extract that actual information. After revival of faulty device, its participation is needed to extract that actual information. This approach promises the security that no device will be able to abuse the information of the faulty device.

##### ***Privacy: Responsibility Re-assignment***

During updating the hash table regarding service re-assignment, the consent of the service provider, a member of that ad-hoc network, is needed, which ensures the user's privacy as well as security.

A comparison table of the existing self-healing and fault tolerance models discussed in chapter three is presented in the next page. Table 7.1 summaries the features supported and not supported by the popular self-healing models-

**Table 7.1. Comparison of Self-healing models**

Model	Infrastructure Support Needed	Transparent	Secure	Privacy Aware	Lightweight	Smart Space Needed	Consider Mobility
Proactive recovery [Pertet04]	Yes	Yes	Yes	N/A	No	No	No
Architectural model [Garlan02]	Yes	Yes	N/A	N/A	No	No	No
Software architecture [Dashofy02]	Yes	Yes	N/A	N/A	No	No	No
AMUN [Trumler04]	N/A	Yes	Yes	Yes	No	Yes	Yes
SHWN [Kant02]	Limited	Yes	Yes	No	Yes	No	Yes
FTA [Tohma02]	Limited	Yes	N/A	N/A	Yes	No	Yes
FDA [Mills04]	No	Yes	Limited	N/A	Yes	No	No
Our Approach	No	Yes	Limited	Yes	Yes	No	Yes

To address all the above characteristics we have designed and implemented an efficient, lightweight, and secure self-healing model targeted for autonomic pervasive computing environment. In the next chapter we will present an evaluation of this model.

## Chapter 8: Evaluation

We have evaluated the performance and usability of our self-healing model by implementing a prototype, designing applications and using simulation tool. An application has been designed that uses our self-healing model. Prototype implementation is described in section 8.1. We have also measured the battery power consumption and the signal strength after using our model that is shown in section 8.2.

### 8.1. Prototype implementation

A prototype including fault detection, fault notification, and faulty device isolation for autonomic pervasive computing environment has been developed and integrated along with our current developed middleware named MARKS+. WinCE running on a set of Dell Axim X30 pocket PCs (Process type is Intel@PXA270, memory size is 160 MB, and speed is 624 MHz), to demonstrate our approach, is used as platform. The .NET Compact framework along with C# is used as implementation language. Bluetooth, as the underlying wireless protocol, has been used though it is also suitable for IEEE 802.11.

Socket and thread programming have been used for successful communication among all the devices in the ad-hoc network. In a healing manager, one thread is used for each device. The thread number is also dynamic due to the ad-hoc nature of the network. An SQLCE database, to store the information, is also used.

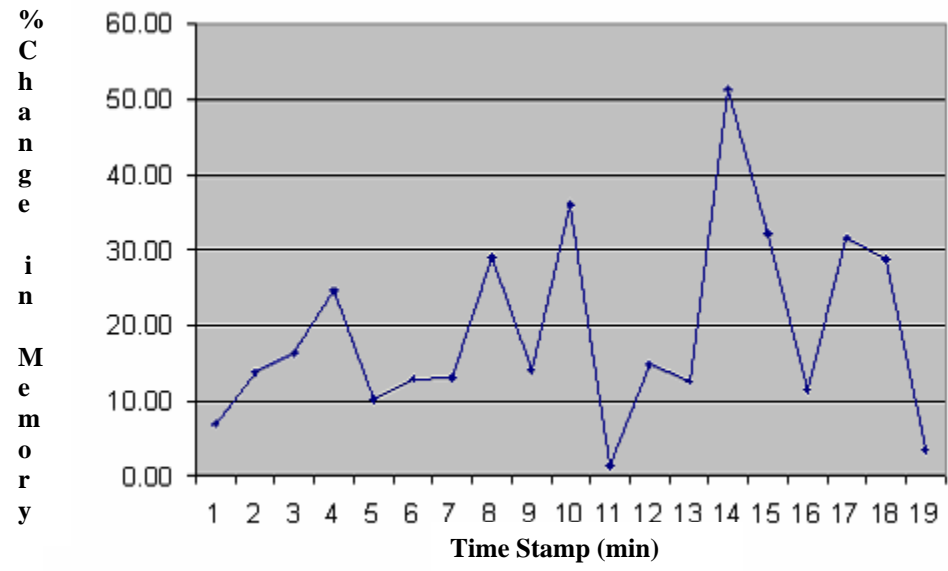
To evaluate the performance of our self-healing model, several applications have been developed. Wireless exam is such an application by which one teacher can send questions to the students (PDA to PDA communication) and also can collect the answers from the students. There are some selected screen shots captured from the implemented prototype below.

Figure 8.1 presents a log file stored in an embedded device (a pocket pc exploited in the application which used the first prototype of fault detection, fault notification, and faulty device isolation). Such a log file is really necessary for a self-healing system.

mem	sp...	mp...	bkp...	ss	app
11	600	97	100	97	5
11	597	91	100	95	5
12	600	97	100	96	7
12	595	97	100	97	5
11	600	95	100	97	8
11	627	97	100	98	9
13	600	97	100	97	10
11	600	92	100	99	6
10	590	96	100	90	9
12	600	97	100	96	7
11	597	91	100	95	5
15	610	90	99	98	6
14	600	97	100	97	10
12	595	97	100	97	5
11	600	95	100	97	8
16	612	98	100	98	9

**Figure 8.1. Status of a device when fault detection process has been applied**

Figure 8.2 illustrates the nature of rate of change of used memory space over time through which self-healing can determine the possibility of a problem between time stamp 14 and 15 due to the sharp change of rate of used memory space.



**Figure 8.2. Rate of change of used memory**

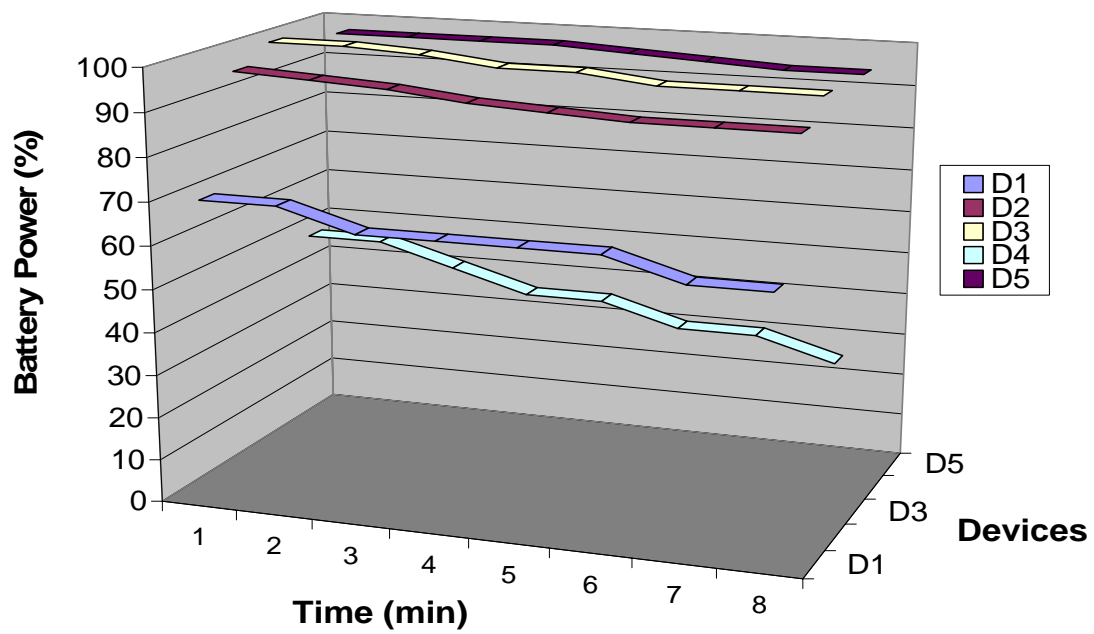


Figure 8.3. Status of the battery power of five devices after using proposed system

Figure 8.3 presents the status of the battery power for five devices where the prototype of our proposed system and MARKS+-ORB are running. The sharp change of the status of the battery power for D4 indicates that there is some problem in that device and needs healing immediately.

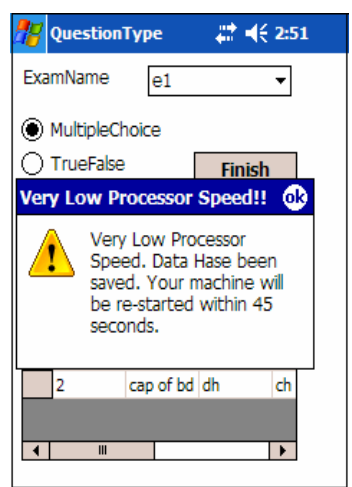
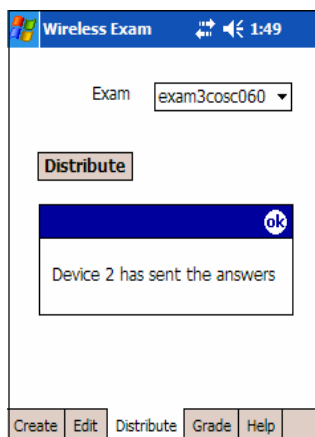


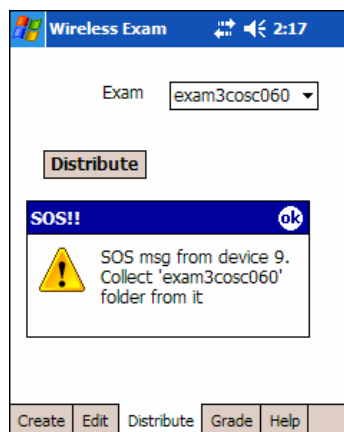
Figure 8.4. Message to the user regarding low processor speed

By using the “status changing rate” process, self-healing of the student’s device itself tries to find out the fault as well as the reason if there is any. Figure 8.4 shows a typical message generated by the devices’ healing unit. This message is intended to inform the user about the abrupt change of device status and action taken by the self-healing.



**Figure 8.5. Device 2 is running without any problem**

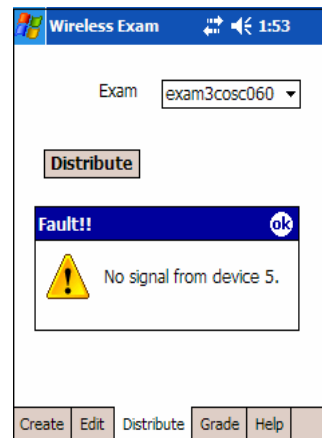
There are situations where all the devices operate without any error. Then only the “OK” message is sent to the healing manager periodically. Suppose device 2 has no problem and it periodically sends OK message to teacher, the healing manager of this network. Within a specified period of time, it also sends all the answers to the teacher’s PDA. This scenario is shown in figure 8.5.



**Figure 8.6. Device 9 sends SOS message**

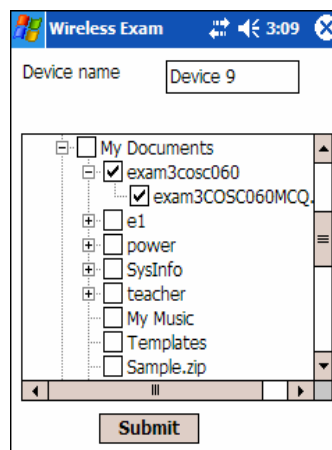
Now device 9 finds some problem. It simply sends SOS (Save Our Soul) message including file name exam3cosc060. Without any delay, the healing manager will collect that file from this device. Figure 8.6 portrays this event.

Another case can occur where the device is unable to send any message due to fault. If the healing manager does not get any message for a long period of time from any device, it will take appropriate action assuming that the device is in fault. Device 5 is unable to send any message for a long time. So, the healing manager takes rapid action regarding device 5. This incident is illustrated in figure 8.7.



**Figure 8.7. No signal from device 5**

As both device 5 and 9 are faulty now, the healing manager removes the entry of device 5 and 9 from hash table. It also updates the table to reassign the services.



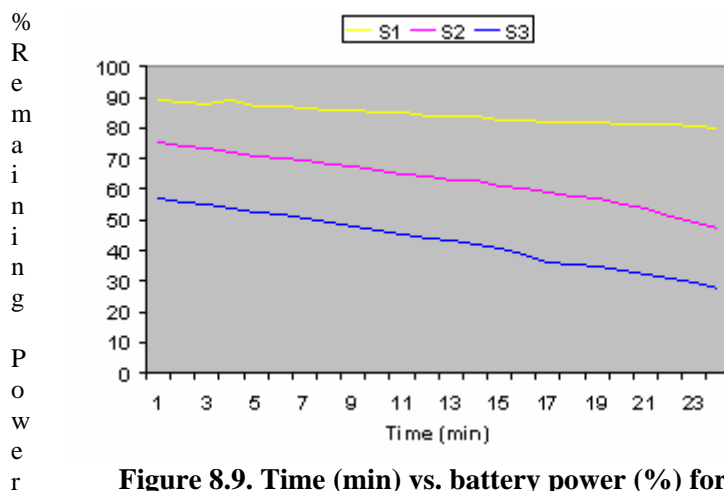
**Figure 8.8. Important files name selected by user to be saved**

Along with the SOS message, device 9 sends the list of important file names (selected by the user of that device) that need to be saved. This is shown in figure 8.8.

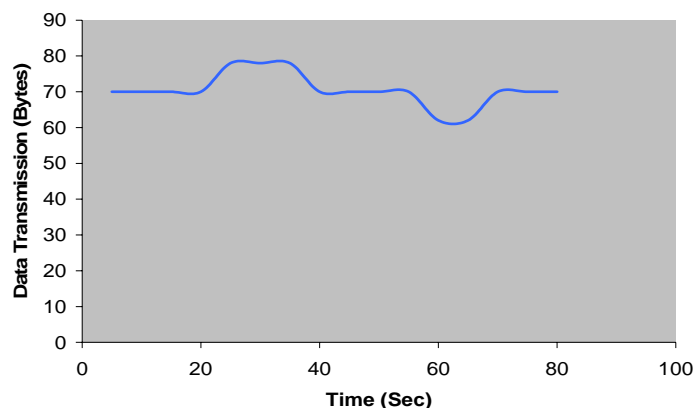
Device 5 and 9 will try to be healed without the help of others. After healing, the healing manager will resend the files that it got before their fault.

## 8.2. Performance measurement

We have also developed the MARKS+-ORB, to provide the device discovery and communication functionality of the devices. Figure 8.9 shows the battery power consumption with respect to time while MARKS+-ORB is running in that device. Here, S1 indicates the battery power consumption while the Pocket PC is on but the wireless mode is off (no wireless communication via 802.11 or Bluetooth). S2 means that wireless mode is on. S3 indicates that wireless mode is on and MARKS+-ORB is running in the device. It clearly indicates that MARKS+-ORB consumes a very little battery power.



**Figure 8.9. Time (min) vs. battery power (%) for MARKS-ORB**



**Figure 8.10. Time (sec) vs. data transmission (Bytes) for MARKS+-ORB**

MARKS+-ORB itself transfers data mainly for device discovery. It broadcasts its own IP address and receives the IP addresses of other devices reside in the same ad-hoc network. Figure 8.10 shows the data transmission by MARKS+-ORB in every 5 seconds. It clearly shows that it does not need to transmit so much data for device discovery.

Table 8.1 shows the line of code and the size of the executable file of Self-healing Service and MARKS-ORB+.

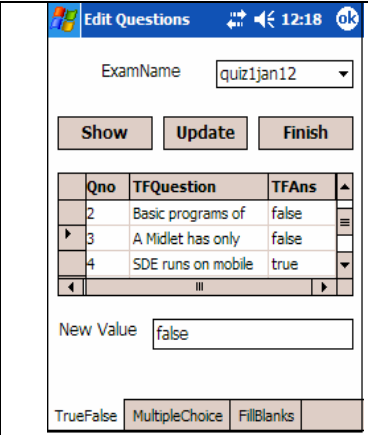
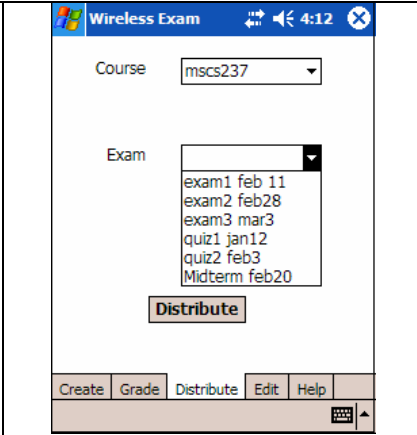
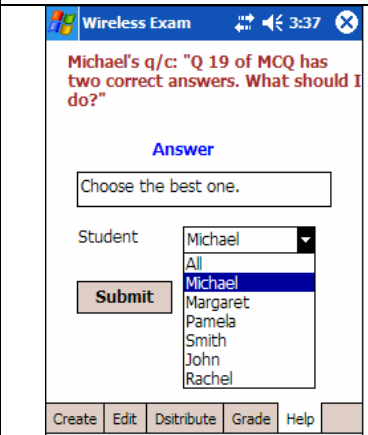
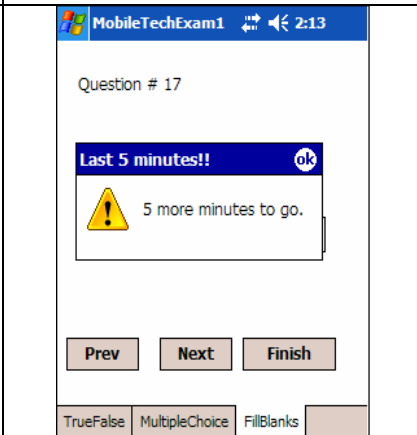
**Table 8.1. Memory footprint for our proposed system and MARKS-ORB**

	Lines of Code	Size of Executable File (KB)
Self-Healing Service	4974	124
MARKS-ORB	416	7
MARKS+	10897	812

### 8.3. Application that uses self-healing model

We have developed some applications that use a self-healing model as a service. We have used applications to check the practicality of using our proposed self-healing model. The overhead of running this service is also tested by running these applications. Some screenshots of the wireless examination application is presented below-

Table 8.2. Screenshot of wireless exam application using our proposed self-healing model

 <p>ExamName: quiz1jan12</p> <p>Show Update Finish</p> <table border="1"><thead><tr><th>Qno</th><th>TFQuestion</th><th>TFAns</th></tr></thead><tbody><tr><td>2</td><td>Basic programs of</td><td>false</td></tr><tr><td>3</td><td>A Midlet has only</td><td>false</td></tr><tr><td>4</td><td>SDE runs on mobile</td><td>true</td></tr></tbody></table> <p>New Value: false</p> <p>TrueFalse MultipleChoice FillBlanks</p>	Qno	TFQuestion	TFAns	2	Basic programs of	false	3	A Midlet has only	false	4	SDE runs on mobile	true	 <p>Course: mscs237</p> <p>Exam: exam1 feb 11, exam2 feb28, exam3 mar3, quiz1 jan12, quiz2 feb3, Midterm feb20</p> <p>Distribute</p> <p>Create Grade Distribute Edit Help</p>
Qno	TFQuestion	TFAns											
2	Basic programs of	false											
3	A Midlet has only	false											
4	SDE runs on mobile	true											
<p>(a) True false questions</p>	<p>(b) Distribute the questions</p>												
 <p>Michael's q/c: "Q 19 of MCQ has two correct answers. What should I do?"</p> <p>Answer</p> <p>Choose the best one.</p> <p>Student: Michael (selected)</p> <p>Submit: Michael, Margaret, Pamela, Smith, John, Rachel</p> <p>Create Edit Distribute Grade Help</p>	 <p>Question # 17</p> <p>Last 5 minutes!!</p> <p>5 more minutes to go.</p> <p>Prev Next Finish</p> <p>TrueFalse MultipleChoice FillBlanks</p>												
<p>(c) Instructors feedback</p>	<p>(d) Auto-generated message</p>												

## Chapter 9: Conclusion and Future Work

In this thesis, we have described the detailed design and implementation of fault detection, fault notification, and faulty device isolation in an autonomic pervasive computing environment. In this chapter, first of all we give a brief summary of our approach to fault detection, fault notification, and faulty device isolation. This summary has been portrayed in section 9.1. The contribution of our thesis has been described in section 9.2. What should be the short term and long term goal of our thesis has been described in section 9.3. Finally, we give some directions for future work in section 9.4.

### 9.1. Summary

We have proposed a solution for fault detection and notification and faulty device isolation in autonomic pervasive computing. The fault detection process promises not only the least possible time to detect the fault but also the lowest degree of user intervention and hence makes our solution highly transparent. We have introduced the concept of healing manager in this regard. Though some researchers have already proposed different solutions for fault-tolerance from a distributed computing and autonomic computing perspective, no solution has been proposed yet for autonomic pervasive computing. Researchers have just started addressing self-healing issues in pervasive computing, let alone autonomic pervasive computing. Our solution is a unique one. We have developed our fault detection, fault notification and faulty device isolation system as a part of MARKS+. MARKS+ is a middleware and the extended version of our previously developed MARKS. MARKS is for pervasive computing and MARKS+ has been developed for autonomic pervasive computing.

## 9.2. Contribution of this thesis

The contribution of this thesis is as follows:

1. Classification of autonomic pervasive computing environment: We have classified this environment into three categories from infrastructure perspective. Some environments have fixed infrastructure, some environments don't have any infrastructure and some environments follow the hybrid approach. We mainly concentrated on the infrastructure less autonomic pervasive computing.
2. Classification of fault: Fault has been defined and classified in different ways in different situation. We have classified from the light of autonomic pervasive computing. We have classified into 4 categories: Hardware Fault, Software fault, Prioritized fault, and Communication Fault.
3. Attributes of self-healing system for autonomic pervasive computing: We have proposed some solutions for fault detection, fault notification and faulty device isolation. These solution have the following attributes:
  - Less Memory footprint: Our system doesn't require much memory
  - Less Time complexity: The overall time complexity of our system is  $O(n)$ , where  $n$  is the maximum number of devices.
  - Transparent: Our system involves nominal user intervention to do fault detection, notification, and faulty device isolation.
  - Solution for Infrastructure less system: Our system concentrates only at infrastructure less system.
  - Scalable system: Even with the increase of the number of the devices in autonomic pervasive computing environment, our solution works fine.

## 9.3. Short and long term impact

Self-healing is a new issue yet very important issue in autonomic pervasive computing. Fault detection and notification are two basic steps to achieve self-healing. Our proposed solution can be used to detect as well as notify a fault of a faulty device. The result of our research will help to accelerate the development of a full-fledged self-healing system for autonomic pervasive computing. It makes the entire fault detection and notification system transparent. We are also planning to make the software freely available through Internet. If the other researchers want to do further research on fault detection and notification, we believe that our software as well as source code will help them. Furthermore, our research results will help others to work on the other parts of self-healing such as alteration, information distribution, fault healing, and selection of healing manager, etc.

We firmly believe that our research potentially can have a great impact on academia and industry in the near future. The use of mobile devices is an integral part of our day to day life. With the increase of the use these devices, self-healing of these devices is becoming more and more important each day. Fault or unusual behavior is very common phenomenon of these mobile devices. Academic courses can be designed to do further research on this topic as the use of mobile and handheld devices are increasing exponentially. Also industry will adopt the system to reduce their maintenance cost.

#### **9.4. Future work**

In future, we will develop a full-fledged self-healing system for autonomic pervasive computing. In our thesis, we have addressed three components of a self-healing system. For a full-fledged self-healing system, the other components need to be addressed. Here, we are giving an outline for the solution of the other parts of self-healing.

*Alteration- Responsibility Re-assignment in other Devices:* Since there might have some devices largely depending upon the faulty device for any specific service, it is also crucial to have some

alternative to continue the smooth functionality. Some middleware like Gaia uses the surrogate device concept for an alternate solution, where it needs to find out another device that is available as well as compatible with the faulty device [Chetan04]. These two fold processes (availability and compatibility) can be performed effortlessly in our Self-healing since the previously mentioned hash tables, regulated and updated by healing manager, shows only those devices which are compatible as well as available.

**Information Distribution:** To assist the faulty device for keeping all the important information safe and secured, the healing manager will distribute the important information (e.g. important database) among other existing devices. To cope with the limited storage capacity of all other devices, secret sharing ( $N, t$ ) approach [SecretSharing] can be used. To preserve the security (make information inaccessible from any intruder or unauthorized user), a modified version of secret sharing might be used. In that case, without the faulty device's consent, the information of that device can't be retrieved by anyone, not even by the healing manager.

**Fault Healing:** The healing manager, in the majority of the circumstances, is not directly associated to healing; rather it assists the device to revamp its previous impossible situations. It stores all the crucial information including log status file of the faulty device when the device falls into trouble. After recuperating from a fault, the healing manager re-collects all information and sends it to that device including log status file so that the device can restore easily its previous condition. In case of failure due to the lack of memory space, the healing manager can help by storing its important but not currently used information among other devices using the modified secret sharing approach [Sharmin05]. The user of the faulty device will have to choose those file names through self-healing.

***Selection of Healing Manager:*** It depends on different criteria like fast processing capability, large available memory space, more functionality and mostly the nature of the applications running among the devices of the ad-hoc network. In our approach, a simple yet powerful process can be used. The main communication point like a server in distributed computing is primarily selected as healing manager. To identify that central point, the status of running application has been checked. The device, which involves the applications mostly, is considered as the healing manager.

***Body Sensor Network:***

One of the most important research areas of Body Sensor Network (BSN) [BSN04, BSN05] is to monitor the patient's physical condition continuously. Since BSN deals with the human life, having a fault tolerant as well as self-healing system is utmost important. We are also planning to use our self-healing system in BSN.

## Bibliography

[Abowd00] G. Abowd and E.D Mynatt, "Charting Past, Present, and Future Research in Ubiquitous Computing," *ACM Trans. Computer Human Interaction*, vol. 7, no. 1, March 2000, pp. 29-58.

URL: [www-static.cc.gatech.edu/fce/pubs/tochi-millennium.pdf](http://www-static.cc.gatech.edu/fce/pubs/tochi-millennium.pdf) (accessed in July 2006)

[Ahamed05] S. I. Ahamed, M. Sharmin, S. Ahmed, M. J. Havice, and S. Anamanamuri, "An Assessment Tool for Out of Class Learning using Pervasive Computing Technologies," *Journal of Information*, vol. 8, no. 5, Sep 2005, pp. 751-768.

[Ahmed05] S. Ahmed, M. Sharmin, and S. I. Ahamed, "Knowledge Usability and its Characteristics for Pervasive Computing Environments," *Proceedings of the 2005 International Conference on Pervasive Systems and Computing (PSC-05) in conjunction with The 2005 International Multi-conference in Computer Science and Engineering*, Las Vegas, NV, USA, June 27-30, 2005, pp. 206-209.

URL: <http://www.mscc.mu.edu/~sahmed02/KnowledgeUsability.pdf> (accessed in July 2006)

[Appavoo02] K. J. Appavoo, M. S. Hui, R. W. Wisniewski, D. D. Silva, O. Krieger, and C. A. N. Soules, "An infrastructure for multiprocessor run-time adaptation," *Proceedings of the first workshop on Self-healing systems*, Charleston, South Carolina, 2002, pp.3-8.

URL: <http://www.lcs.ece.cmu.edu/~soules/papers/woss.pdf> (accessed in May 2006)

[Blair02] G. S. Blair, G. Coulson, L. Blair, H. D. Limon, P. Grace, and R. M. N. Parlavantzas, "Reflection, self-awareness and self-healing in OpenORB," *Proceedings of the first workshop on Self-healing systems*, Charleston, South Carolina, 2002, pp.9-14.

URL: <http://www.lancs.ac.uk/postgrad/parlavan/publications/blairwoss02.pdf> (accessed in May 2005)

[Bobbio98] A. Bobbio and M. Sereno, "Fine grained software rejuvenation models," *Computer Performance and Dependability Symposium (IPDS 98)*, September 1998, pp. 4–12.

[Bracewell03] T. D. Bracewell and P. Narasimhan, "A Middleware for Dependable Distributed Real-Time Systems," *Joint Systems and Software Engineering Symposium*, Falls Church, VA, April 2003.

URL: <http://www.ece.cmu.edu/~mead/raytheonSymp-2003.pdf> (accessed in May 2006)

[Broch98] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva, "A Performance Comparison of Multi-Hop Wireless Ad-Hoc Network Routing Protocols," *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, Dallas, Texas, October 1998, pp. 85-97.

URL: <http://monarch.cs.rice.edu/monarch-papers/mobicom98.pdf> (accessed in July 2006)

[Brown05] A. B. Brown and C. Redlin, "Measuring the Effectiveness of Self-Healing Autonomic Systems," *Proceedings of the Second International Conference on Autonomic Computing (ICAC'05)*, June 13-16, 2005, pp. 328-329.

URL: [csdl.computer.org/comp/proceedings/icac/2005/2276/00/22760328.pdf](http://csdl.computer.org/comp/proceedings/icac/2005/2276/00/22760328.pdf) (accessed in July 2006)

[Chetan04] S. Chetan, A. Ranganathan, and R. Campbell, "Towards fault tolerant pervasive computing," *Pervasive 2004 Workshop on Sustainable Pervasive Computing*, Linz /Vienna, Austria, April 2004.

URL: <http://choices.cs.uiuc.edu/~chetan/papers/tfpc.pdf> (accessed in May 2006)

[Dashofy02] E. M. Dashofy, A. V. D. Hoek, and R. N. Taylor, "Towards architecture-based self-healing systems," *Proceedings of the first workshop on Self-healing systems*, Charleston, South Carolina, 2002, pp. 21-26.

URL: <http://www.ics.uci.edu/~andre/papers/C23.pdf> (accessed in May 2006)

[Ganek03] A. G. Ganek and T. A. Corbi, "The dawning of the autonomic computing era," *IBM Systems Journal*, vol. 42, no. 1, pp. 5-18.

URL: [www.item.ntnu.no/fag/ttm4128/lectures/The-dawning-of-the-autonomic-computing-era.pdf](http://www.item.ntnu.no/fag/ttm4128/lectures/The-dawning-of-the-autonomic-computing-era.pdf) (accessed in June 2006)

[Garg98] S. Garg, A. van Moorsel, K. Vaidyanathan, and K. Trivedi, "A methodology for detection and estimation of software aging," *International Symposium on Software Reliability Engineering*, Nov 1998, pp. 283–292.

URL: <http://portal.acm.org/citation.cfm?coll=GUIDE&dl=GUIDE&id=856162> (accessed in July 2006)

[Garlan04] D. Garlan, V. Poladian, B. Schmerl, and J. P. Sousa, "Task-based Self-adaptation," *Proceedings of the ACM SIGSOFT 2004 Workshop on Self-Managing Systems (WOSS'04)*, Newport Beach, CA, October-November 2004.

URL: <http://www-2.cs.cmu.edu/~able/publications/WOSS04/> (accessed in May 2006)

[Garlan02] D. Garlan, and B. Schmerl, "Model-based adaptation for self-healing systems," *Proceedings of the first workshop on Self-healing systems Charleston*, South Carolina, November 18-19, 2002, pp. 27- 32.

URL: <http://hdep.org/Publications/WOSS02.pdf> (accessed in May 2006)

[Horn01] P. Horn, *Autonomic Computing: IBM's Perspective on the State of Information Technology*, IBM Corporation, October 15, 2001.

URL: [http://www.research.ibm.com/autonomic/manifesto/autonomic\\_computing.pdf](http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf) (accessed in June 2006)

[Huang95] Y. Huang, C. Kintala, N. Kolettis, and N. Fulton, "Software rejuvenation: Analysis, module and applications," *International Symposium on Fault-Tolerant Computing*, Pasadena, CA, June 27-30, 1995, pp. 381–390.

URL: <http://swig.stanford.edu/~candea/teaching/cs444a-fall-2003/readings/huang-rejuvenation.pdf> (accessed in July 2006)

[Kant02] L. Kant, "Design and performance modeling & simulation of self-healing mechanisms for wireless communication networks," *Proceedings of the 35th Annual Simulation Symposium*, April 2002, pp. 35-42.

URL: <http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/proceedings/&toc=comp/proceedings/ss/2002/1552/00/1552toc.xml&DOI=10.1109/SIMSYM.2002.1000081> (accessed in July 2006)

[Mills04] K. Mills, S. Rose, S. Quirolgico, M. Britton, and C. Tan. "An Autonomic Failure-Detection Algorithm", *Proceedings of the 4th International Workshop on Software Performance (WoSP 2004)*, January 14-16, 2004, San Francisco, California, ACM Press, pp. 79-83.

URL: [w3.antd.nist.gov/pubs/WoSP-Paper8v3.pdf](http://w3.antd.nist.gov/pubs/WoSP-Paper8v3.pdf) (accessed in July 2006)

[Pertet04] S. Pertet and P. Narasimhan, "Proactive Recovery in Distributed CORBA Applications," *IEEE Conference on Dependable Systems and Networks (DSN)*, Florence, Italy, June 2004.

URL: <http://www.ece.cmu.edu/~mead/dsn-2004.pdf> (accessed in May 2006)

[Poladian06] V. Poladian, J. P. Sousa, D. Garlan, B. Schmerl, M. Shaw, "Task-based Adaptation for Ubiquitous Computing," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Special Issue on Engineering Autonomic Systems*, vol. 36, no. 3, May 2006.

URL: <http://www.cs.cmu.edu/afs/cs/project/able/ftp/Submissions/task-ubiquitous.pdf> (accessed in July 2006)

[Roman02] M. Roman, C. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt, "Gaia: A Middleware Infrastructure for active spaces," *IEEE Pervasive Computing*, October-December 2002, pp. 74-83.

URL: <http://choices.cs.uiuc.edu/~ranganat/Pubs/gaia.pdf>, (accessed in May 2006)

[Satya95] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing," *ACM Symposium on Principles of Distributed Computing, 1995 (PODC'95 invited lecture)*.

URL: <http://www-2.cs.cmu.edu/afs/cs/project/coda/Web/docdir/podc95.pdf> (accessed in May 2006)

[Schneider84] F. B. Schneider, "Byzantine Generals in Action: Implementing Fail-Stop Processors," *ACM Transactions on Computer Systems*, vol. 2, no. 2, May 1984, pp. 145-154.

URL: <http://www.cs.virginia.edu/~jck/cs651/papers/fail.stop.in.action.pdf> (accessed in May 2006)

[Secret Sharing] Secret Sharing,

URL: [www.cmpe.boun.edu.tr/courses/cmpe471/spring2003/download/cmpe47109-2003.ppt](http://www.cmpe.boun.edu.tr/courses/cmpe471/spring2003/download/cmpe47109-2003.ppt)

(accessed in May 2006)

[Sharmin05] M. Sharmin, S. Ahmed, and S. I. Ahamed, "SAFE-RD (Secure, Adaptive, Fault Tolerant, and Efficient Resource Discovery) in Pervasive Computing Environments,"

*Proceedings of the IEEE international Conference on Information Technology (ITCC 2005)*, Las Vegas, NV, USA, April 4-6, 2005, pp. 271-276.

URL: <http://www.mscs.mu.edu/~sahmed02/SAFE-RD.pdf> (accessed in July 2006)

[Sharmin06] M. Sharmin, S. Ahmed, and S. I. Ahamed, "MARKS (Middleware Adaptability for Resource Discovery, Knowledge Usability and Self-healing) for Mobile Devices of Pervasive Computing Environments," *Third International Conference on Information Technology: New Generations (ITNG 2006)*, April 2006, Las Vegas, Nevada, USA, pp. 306-313.

URL: [www.mscs.mu.edu/~ubicomp/marks.pdf](http://www.mscs.mu.edu/~ubicomp/marks.pdf) (accessed in July 2006)

[Tohma02] Y. Tohma, "Fault tolerance in autonomic computing environment," *2002 Pacific Rim International Symposium on Dependable Computing (PRDC'02)*, December 16-18, 2002, pp. 3-6.

URL: <http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/proceedings/&toc=comp/proceedings/prdc/2002/1852/00/1852toc.xml&DOI=10.1109/PRDC.2002.1185612>

(accessed in July 2006)

[Tony05] T. O'Donnell, D. Lewis, and V. Wade, "Intuitive Human Governance of Autonomic Pervasive Computing Environments," *Proceedings of the 1st International IEEE WoWMoM Workshop on Autonomic Communications and Computing*, Giardini Naxos, Italy, June 13, 2005, pp. 532-536.

URL: <https://www.cs.tcd.ie/Tony.ODonnell/publications/ACC2005.pdf> (accessed in June 2006)

[Trumler04] W. Trumler, J. Petzold, F. Bagci, and T. Ungerer, “AMUN – An Autonomic Middleware for the Smart Doorplate Project,” *System Support for Ubiquitous Computing Workshop at the Sixth Annual Conference on Ubiquitous Computing (UbiComp 2004)*, Nottingham, England, September 7, 2004.

URL: [http://ubisys.cs.uiuc.edu/proceedings\\_04/amun.pdf](http://ubisys.cs.uiuc.edu/proceedings_04/amun.pdf) (accessed in May 2005)

[Weiser93] M. Weiser, “Some Computer Science Problems in Ubiquitous Computing,” *Communications of the ACM*, vol. 36, no. 7, July 1993, pp. 75-84.

URL: <http://www.ubiq.com/hypertext/weiser/UbiCACM.html> (accessed in May 2006)

[Yau02] S. Yau, Y. Wang, and F. Karim, “Development of Situation-Aware Application Software for Ubiquitous Computing Environments,” *Computer Software and Application Conferences*, 2002, pp. 233-238.

URL: [www.eng.auburn.edu/~yuwang/Compsac02-pres.pdf](http://www.eng.auburn.edu/~yuwang/Compsac02-pres.pdf) (accessed in July 2006)

[Yujuan03] B. Yujuan, S. Xiaobai, and K.S. Trivedi, “Adaptive software rejuvenation: Degradation model and rejuvenation scheme,” *International Conference on Dependable Systems and Networks*, June 2003, pp. 241–248.

URL: [http://ieeexplore.ieee.org/xpl/freeabs\\_all.jsp?isnumber=27228&arnumber=1209934&count=82&index=24](http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?isnumber=27228&arnumber=1209934&count=82&index=24) (accessed in July 2006)

## Appendix A

### Table Definitions

<b>Term</b>	<b>Definition</b>
Pervasive computing	A trend toward an environment where information can be accessible to users anywhere and anytime.
Autonomic Computing	An approach for designing computing systems capable of reconfiguration, fault detection and repair, and performance optimization.
Self-healing	Process of discover, diagnose, and react to faults.
Fault Detection	Process of discovering failure.
Fault Notification	Process of notifying other devices about failure.
Ad hoc network	A temporary network whose members are wireless mobile devices.
Smart Space/Intelligent Environment	A space that has embedded devices and networks to support user tasks.
MARKS	Middleware designed for devices running in pervasive computing environment.
MARKS+	Middleware designed for devices running in autonomic pervasive computing environment.
Healing Manager	Unit of MARKS and MARKS+ responsible for healing process.
Service Manager	Unit of MARKS and MARKS+ that acts as backup for healing manager.
MARKS-ORB	Object Request Broker of MARKS and MARKS-ORB. It is responsible for device discovery and communication process.

## Appendix B

The code of the prototype implementation is available at

<http://www.mscs.mu.edu/~ubicomp/shameem/thesis>