DESIGN AND IMPLEMENTATION OF MARKS (MIDDLEWARE ADAPTABILITY FOR RESOURCE DISCOVERY, KNOWLEDGE USABILITY AND SELF-HEALING) MIDDLEWARE FOR PERVASIVE COMPUTING ENVIRONMENTS*

Moushumi Sharmin, Shameem Ahmed, and Sheikh I. Ahamed
Marquette University, Milwaukee, Wisconsin, USA
Contact author: iq@mscs.mu.edu

ABSTRACT

According to a recent survey, in 2005, 500+ million people used handheld devices (PDA, cell phone, etc.) worldwide, while this number was 200+ million in 1999[33]. Hence, this increase can play a vital role for pervasive computing. The goal of pervasive computing is to provide services to anyone anytime, overcoming the constraints of place, time and character. With this tremendous increase of use of handheld and wearable devices, the pervasive computing arena is becoming more strong and powerful day by day. Despite of having various physical constraints, most of the facilities enjoyed by resource-rich devices are tried to be incorporated in these tiny devices. That is why different research fields have been developed to cover this area. There are still some unexplored but crucial attributes like Knowledge Usability, Resource Discovery, and Self-healing that deserve high attention. Besides exploration, implementation as well as evaluation of these features from security and privacy perspectives is also needed. In this paper, we illustrate the design and implementation of a middleware, which incorporates these less explored areas of pervasive computing not only to guarantee the optimum utilization of the physical capabilities but also to ensure the highest degree of security and supreme privacy. The name of the middleware is MARKS since it stands for Middleware Adaptability for Resource Discovery, Knowledge Usability and Self-healing for Pervasive Computing Environments, which are illustrated in this paper.

Keywords: Pervasive computing, Mobile computing, MARKS, Mobile devices, Middleware for mobile computing, Middleware for pervasive computing

1. INTRODUCTION

Albeit the exponential growth of exploitation of handheld devices, these devices themselves are suffering from a number of limitations [1] to date, which includes but not limited to inadequate processing capability, restricted battery life, limited memory space, slow expensive connections, frequent line disconnection, and confined host bandwidth. To cope with the ever-growing requirements, middleware can play an essential role. A number of middleware have been designed to deal with these challenges but no single one can be considered the ultimate solution.

Applications running in a pervasive/ubiquitous/mobile computing environment focus primarily on context-sensitivity, situation-awareness, and ad-hoc communication [2, 3]. The ability of a device to sense its present context and changes in contextual data is called Context-sensitivity; a device's capability not only to capture but also to analyze the relationship among multiple contexts and actions over a period of time is known as situation-awareness; with the change of contexts, mobility of devices and availability of resources, the instantaneous establishment and termination of communication channels among applications is called ad-hoc communication.

The motivation of pervasive computing is to force the computer and computing facilities to live with people in a conjunctive manner. This makes it an extremely difficult integration of human factors, computer science, engineering and also social sciences [4]. This necessitates the devices to become an inseparable part of the user's daily life. Another aspect that makes pervasive computing unique is its goal to provide transparent services. This transparency necessitates the proper knowledge about user preferences and the specific behavior that the user expects in an individual circumstance. In fact, context-sensitivity, situation-awareness, and ad-hoc communication didn't address till the analysis of information to decide the appropriate action during a particular situation from application's point of view. Information about user and its analysis depending on users past behavior and the changing situation is not addressed by context-sensitivity or situation-awareness. Pervasive computing demands a new thread, which should be appended as the fourth characteristic of pervasive computing applications. This new attribute, termed "Knowledge Usability" [5], is properly handled in our designed middleware. In short, our middleware will be a guide and friend of the user in each moment of his/her life.

*A short version of this paper has been presented in *the Third International Conference on Information Technology: New Generations (ITNG 06)*

Resource Discovery is an integral part of every system running in a pervasive computing environment [6]. This process optimally explores a device capable of offering a specific resource. Typical devices operating in this environment are resource poor. Consequently, the dependency on other devices for specific resources is a very common phenomenon. But due to the ad hoc and ephemeral nature of the network, one can't expect to get service from a particular device for a long span of time. Moreover, multiple devices may concurrently request the same specific service. These aspects demand a scalable, efficient, quickly responsive, and dependable middleware that meets all the needs of devices running in a pervasive computing environment. At present, research [3, 7, 8] on middleware for pervasive computing does not offer any complete solution from Resource Discovery, Knowledge Usability and self-healing standpoint. Therefore, a novel approach, to facilitate this quest, is very needed which efficiently provides services to many devices concurrently, reduces the seeking time, and also tracks the other devices as substitute resource providers [9]. In this paper we are presenting our middleware "MARKS" that takes care of all of the above issues of service discovery using a *cluster based dynamic hash algorithm*.

A fault tolerant system, an indispensable part of self-healing, requires that the system will continue its operation even after an occurrence of any type of fault, predictable or unpredictable [10]. To maintain the consistency of application data in the event of a fault is the primary characteristic of the above system. The recovery method, which includes application level perseverance of data, identifies the set of actions that should be taken to address assumed failures [11]. The self-healing property addresses fault detection, resource recovery, and fault healing. This system follows the principle that faults can occur unexpectedly, and should be handled using a recovery mechanism commenced by the faulty system itself. Proactive fault prediction, followed by pertinent act to recover will help to lessen the recovery time and smooth the functioning of devices. Significant works have been done in the area of fault-detection and recovery in distributed real-time systems and autonomous systems. But in the pervasive computing environment, any noteworthy approach is yet to be developed. G. Banavar et al. [12] point out the need to modify the conventional fault detection and recovery techniques. Our objective is to integrate the capability of fault detection, resource recovery, and healing at the middleware level. The probable faulty devices' middleware will instigate the actions needed to recover by a *rate of change of status* process. It will be followed by fault notification, device isolation, alteration, information distribution, and fault healing [13].

The development of pervasive middleware requires more emphasis on Resource Discovery, Knowledge Usability, and self-healing. To fulfill this aim, in this paper we are presenting the design and implementation of MARKS (Middleware Adaptability for Resource Discovery, Knowledge Usability and Self-healing), which is an adaptive middleware that supports Resource Discovery, Knowledge Usability and Self-healing capability. Here we introduce the architecture and implementation issues of MARKS. We discuss the components of MARKS and there operation in details. We also discuss about the security and privacy features supported by MARKS. We introduce the concept of resource manager and healing manager that enhances the dependability of the middleware services.

Security and privacy are two major concerns in the area of pervasive computing. The sharing of resources among many devices belonging to an ephemeral group makes it more difficult to ensure security. As users are sharing their resources, it is highly important to respect their privacy. At the same time, it is the system's own responsibility to guarantee that the resource provider is not adversely affected. In our middleware, we consider these crucial aspects and try to maintain both privacy and security by implementing *modified secret sharing* [14] and *providers consent method* [9].

In section 2, we present several scenarios describing the necessity of incorporating the above features in a middleware. We illustrate the description of our approach along with our algorithm in section 3. The architecture of each unit is described along with diagrams in this section. We also demonstrate the implementation and evaluation of our approach in section 4. Section 5 deals with the current state of the art. We present our future research direction, and concluding remarks, in section 6.

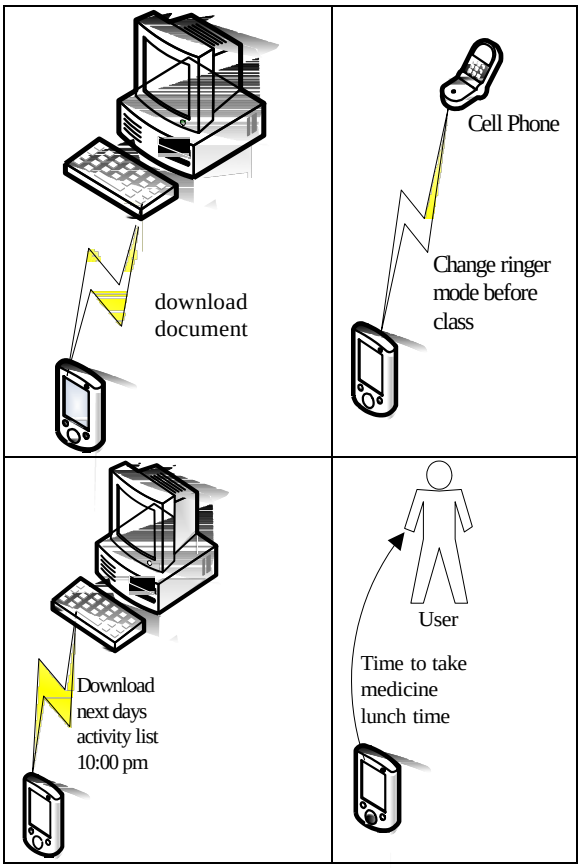

Figure 1. Use of knowledge usability service

2. MOTIVATION

MARKS can aid a user in different aspects of his life, provided the entire system consists of mobile devices having wireless communication capacity and data exchange competence among them, which include but are not limited to PDAs, laptops, or smart phones. Other non-mobile yet powerful devices (e.g. desktop PC), if available, can accelerate the job and can assist these mobile devices.

Scenario 1

Dr. Chambers, a history teacher, has to deliver a lecture on alternate days at 10:00 am in his HIST 235 class. Before leaving from home for the class, all the necessary presentation documents should be uploaded on his PDA. An application based on MARKS, the middleware running on Dr. Chamber's device, will initiate the uploading process just in time without his physical intervention.

MARKS also eases Dr. Chamber's job by automatically changing the ringer mode on his cell-phone to another appropriate mode. MARKS will also not forget to restore the ringer mode when the lecture is finished.

During lunch, MARKS not only informs him of his remaining schedule for the day but also notifies him to take prescribed medicine.

On his way home, after checking his "to do" list, MARKS will inform him that he needs to buy groceries. Not only that, after collecting the contextual information (e.g. the location of Dr. Chambers), MARKS will consult the databases for item availability as well as its price at the nearby shops. After analyzing all this relevant information, it will suggest Dr. Chambers go to a particular shop, which best matches his preference. Before going to bed, MARKS will summarize the next day's activities. Figure 1 depicts this scenario.

MARKS, in a nutshell, will act as a guide or personal assistant in a non-intrusive manner so that the user (here Dr. Chambers) will be minimally disrupted.

Scenario 2

Lydia goes to a career fair to collect all necessary information about getting a job. At the fair, she finds representatives from some companies have their own booths and are delivering short speeches regarding the employment criteria, nature of work etc. that seems very important to her. Since Lydia does not have any prior experience with attending career fairs, she has not brought any recording software. MARKS, the middleware running on Lydia's PDA, requests resource manager (RMX), the MARKS of John's PDA, for recording software. RMX assigns the recording software of Michael's PDA regarding this request. After a while, John leaves the booth and consequently RMX allocates another device for serving Lydia's request. Lydia has finished her visit in the booth and moves to a new booth. As she joined a new cluster, the resource manager of this cluster (let RMY), offers her the same functionality using a suitable device.

Scenario 3

During a boat trip on Lake Michigan to collect data from lake water, a group of marine biology students wanted to share their sensor data (stored in their handheld devices) to enrich their knowledge. During their information exchange, the healing manager of the cluster detects that one of the devices is having a high probability of going down.

To avoid the loss of data stored in that faulty device, the healing manager will disseminate the stored information to the remaining devices. Consulting the logbook, necessary measures will be taken to restore the devices prior working state. Disseminated information content will be refurbished to the device, to help it work to its fullest capacity.

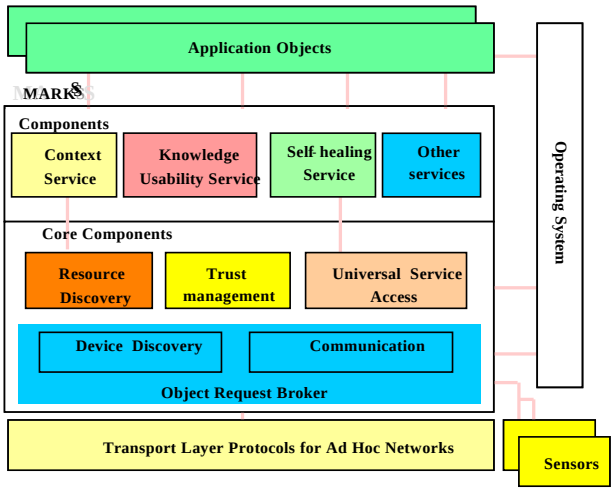

Figure 2. MARKS architecture

3. OUR APPROACH

Our middleware MARKS is composed of core components and services. Core components include the object request broker (ORB), Resource Discovery, trust management and universal service access unit. The ORB handles the device discovery and communication part of MARKS. On top of the core components the services offered by MARKS sits. The services currently supported are context-service, Knowledge Usability service, and self-healing service. There is also room for adding more services. Application objects communicate with services and service components communicate with ORB. The communication to other devices is done by ORB. Figure 2 depicts the MARKS architecture. Details of Knowledge Usability, Resource Discovery and Self-healing units are presented in the following subsections:

3.1. Knowledge Usability

The small foot print memory of embedded devices poses a great challenge in accumulating the knowledge among the distributed devices in a close proximity. It is of great importance to mine past, present, and future user information, to determine its necessity. Moreover, a framework is needed to address *accessibility, consistency, transparency, robustness, and security* of knowledge.

Notion of Knowledge Usability: Knowledge is a set of activity states (α_1, α_2, α_3, α_4,..) of the user (U) where α_i (i>=1) represents U's preferences. The term Knowledge Usability refers to the autonomous technique to determine α_i and also analyze it to decide U's appropriate action at a particular time [5].

Information Mining and Storing: Depending on user activities of the past, present and future (consulting with a "to do" list or task calendar), a weekly task pattern is devised. A pattern is retrieved by means of a simple matching algorithm. To cope with the memory constraint, the mobile devices in a pervasive computing environment should have some optimization capability to accumulate all the necessary information. For that purpose, we designed a *matrix based knowledge dissemination approach.*

The dimensions of the two-dimensional matrix represent the time frame of each day and days of the week. Each cell of the matrix lists a specific user task. A standard is used to assign a

number for each typical activity. The irregular task (user dependent) is allocated some other pre-specified number. The concept of number is introduced to guarantee the best possible use of the storage capacity. As an example, the regular tasks such as going to lunch, attending a meeting or buying groceries have a number like 1, 2, 3, and so on. Taking piano lessons, a very user centric activity, would have some pre-specified number like 500. Mapping of numbers with activities is distributed to all devices of close proximity. It ensures the optimum storage utilization of all the devices. Also, the matrix can be dispersed, if very large, among devices residing in the same cluster. Change in the number of devices can dynamically reallocate the matrix contents.

3. 2. Resource Discovery

For discovering resources in a close proximity "*Cluster based Dynamic Hash Algorithm*" is used [9].

The devices in a close proximity form tree-like clusters. Each tree is a height one general tree. To search a service, according to conservative style, at first, a device, investigates its own cluster through service manager, which will be treated as the root of the tree. It also reflects its "*Proximity Awareness"* property, which means the preference is for local service first. Unavailability of such a service leads to exploration of the other clusters in the forest. It ensures the most minuscule searching time; making MARKS a quickly responsive middleware.

"*Modified Announcement"* [9] is used instead of "*query process"* [15] for service look-up. "*Modified Announcement"* policy follows parent-child relationship to announce available services. Devices declare their resource information through their parent to the service manager to help it accumulate all the available service information in it.

To use the tiny storage capacity efficiently, some numbers are used for representing these services. To reduce the searching time, a hash table is used for resource searching. Time complexity for this is O (1), i.e. constant time. Upon success, the resource manager provides the service facility to the device. Otherwise, the resource manager consults with other resource managers of other clusters residing in that forest. The detailed description of the process along with algorithms can be found in [9].

For "*Resource matching"*, the Resource Manager of MARKS uses the "*Match all"* technique. However for "*Resource Selection", the "Match best"* approach is followed. Populating the hash table with resource information follows "*Match all"* procedure. Resource query follows "*Match Best"* approach, followed by "*Next Best Match*" in case of multiple requests for the same resource. MARKS can ensure, by this sequence, to provide the best available services to the most number of devices. No middleware can be claimed as capable as MARKS.

Security and privacy issues are taken care of by adopting "*Providers Consent Method*" [9]. This method ensures that without the explicit approval of the provider, no resource owned by him will be shared. If he wants to share his resources among some of the network members, the resource manager ensures that also. To maintain security, a twofold mechanism is adopted. The resource manager makes sure that any entering network member who wants to have any resource from any existing member is not a threat to the security. A device that can have negative impact on any existing device is considered to be a threat. We are working on a trust model so that a new device can request a service and is not a threat. After that, the resource manager communicates to the specific resource provider to obtain his approval for sharing resource. If the resource provider is not willing to offer his resource to this particular member, then the resource manager looks for an alternate resource provider. Though we want the entire system to be transparent to the user, the precise permission is needed to make the sharing process secure, in particular in case of resource sharing with a device for the first time. MARKS is dependable because, in most cases, it will never fail to provide service. To ensure this, a "*back up strategy*" is followed. Also, to cope with the ad hoc and ephemeral nature of the networks in a pervasive computing environment, no static resource manager is used. A flexible benchmark is deployed to choose the resource manager dynamically. The most eligible device, according to benchmark, carries out the functionality of resource manager and choose the second most qualified one as its running mate, which acts as a backup for its resource related information. This running mate will fill the vacuum created by the first's failure to carry out the task. The next one, if needed, will be chosen accordingly. By this way, MARKS ensures the continuous Resource Discovery support, which makes it a dependable middleware from a resource providing perspective. "*Modified Secret Sharing*" [9] is used to disseminate resource information to avoid single point failure.

To facilitate more complex services to the user, dynamic service integration is included in our Resource Discovery scheme. A cellular automata concept is employed to adopt this feature.

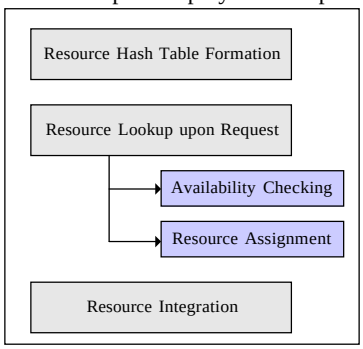

Figure 3. Resource discovery Unit

Figure 3 depicts the architecture of the Resource Discovery unit. When any device joins the ad-hoc network, it is the responsibility of the resource manager to include this new device's information in the hash table. In case of any request for any resource, the resource manager checks for availability and also ensures that the owner of the resource has no objection to sharing. The resource-assignment subunit is responsible for privacy and security maintenance. In case of a request for a particular resource that is not available directly, but can be obtained through the integration of two or more resources, then the resource-integration subunit handles this.

3.3. Self-Healing

The self-healing unit of MARKS is termed as ETS (Efficient, Transparent, and Secured) Self-healing [13]. The concept of a healing manager has been introduced in ETS Self-healing to deal with any kind of fault issues. The resource manager will act as the repository of information of the healing manager for backup purposes.

To ensure supreme quality, the ETS Self-healing unit exploits the process named *rate of change of status* of each device, to predict the fault and two different types of messages ("Ok" and "SOS") to announce the condition to the healing manager. It

periodically monitors, as well as assembles, the status of all of the running applications, memory, power, communication signal etc. Abrupt alteration in those values indicates the possibility of fault. So the device's self-healing unit can detect fault by analyzing the rate of change of these parameter values.

To notify the healing manager about the condition of a device, each one periodically generates "OK" message. But if the device's self-healing unit can predict the upcoming fault, then it sends an "SOS" message with the logstatus file and the important file names that need to be saved in case of any collapse. The healing manager also queries periodically about the messages, and if it gets any "SOS" message or no message at all from a particular device for a long time, then it initiates the recovery procedure.

Resource recovery, the next step to self-healing, has a fourfold processes. "*Fault containment*" [16], the isolation of faulty devices from the remaining cluster, is simply achieved here by removing the service number from corresponding position of the hash table. Time complexity for performing both actions is O (1). The overhead, to maintain this time complexity, is negligible here. In ETS' Self-healing unit, this is done by the "Device Isolation" subunit.

After being informed about the fault of a specific device, as well as its possible causes, the healing manager will explore the alternate solution for the devices that are currently taking service from that faulty device. This step is termed an "*Alteration*". Some middleware like Gaia uses surrogate device [16] concept for an alternate solution, which is incorporated in MARKS by looking up the service provider's number in the hash table.

Information distribution is a *significant attempt at resource recovery,* to revive the proper functioning. Due to the scarcity of storage scope, the healing manager disperses the information content among the working members of the cluster. The amount of information distribution dynamically changes according to the ephemeral nature of the system.

The healing manager consults the logbook, maintained by the defective device, to help it resume its functionality by providing the saved information among the devices. The architecture of the self-healing unit is presented in Figure 4.

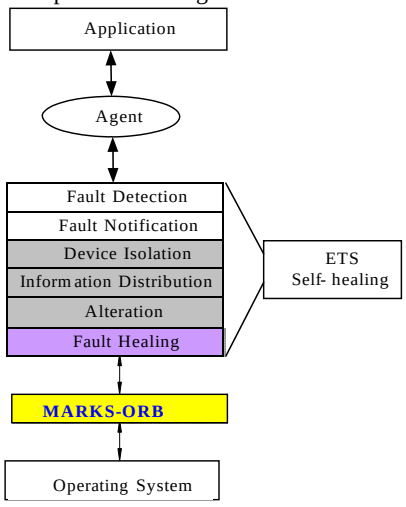

Figure 4. Self-Healing unit architecture

The process of self-healing is achieved through the combined effort of the faulty-device's Self-healing unit and the healing manager of that particular ad-hoc network. The fault detection and notification phases are initiated and operated by each device's ETS Self-healing unit. The fault-healing unit works as a combined effort of the device's Self-healing unit and also the self-healing manager. But the other three phases of the recovery mechanism are controlled exclusively by the healing manager. As the healing manager contains the available resource information of that network, the search for alternative resource-provider (alteration) and information distribution of faulty devices among other members of that network, is done efficiently by it.

The ETS Self-healing unit is generic, so it is able to heal not only application faults but also other kind of faults. It is efficient in terms of speed and space. *Transparency* is also another desirable feature of any healing system. Since both resource manager and healing manger look out for the entire process, then user intervention is not required here; this abides by the *Transparency* rule. Security is also preserved in every procedure of the self-healing unit [13].

4. MARKS: IMPLEMENTATION AND EVALUATION

We have implemented the first prototype of MARKS based on the design shown in figure 4 on Dell Axim 30 pocket PCs. We have used Visual Studio .NET compact framework. In a pervasive computing environment, most of the devices are laptops, PDAs and smart phones. We envision the future-computing environment to be full of these types of smart and small devices. The framework that .NET provides is accessible from applications developed in C, C++, C#, VB and other programming languages. So the sharing of resources and the communication aspect become simple.

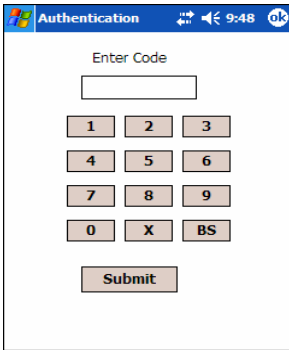

Figure 5. Authentication of the user

One of our major goals is to make MARKS secure. To enhance the security feature of MARKS, we have added an authentication facet on top of its core services. If any application wants to communicate with MARKS or access functionalities provided by it, it needs to authenticate itself. Once the application successfully validates itself as a genuine user, it can access the services without any further security check. The user authentication process is accomplished with the help of security code, which is a combination of numbers. This authentication design was chosen to make it as simple as possible. Figure 5 presents the authentication screen.

We have implemented the privacy issues according to the providers' consent method. When the user joins any ad-hoc

network, the list of resources available to its device is shown to the resource manager and with the users approval, is recorded in the list of available resources.

The first prototype of MARKS included device discovery, service discovery, and self-healing services. To evaluate these units, some applications like a wireless exam scheme and an online survey system, etc., have also been implemented and tested using several PDAs, which runs our MARKS.

In the case of a wireless exam application, a teacher can send different types of questions (multiple choice, True/False, fill in the blanks, and so on) to the students (PDA to PDA communication). After taking the exam, students will also be able to send their answers, which would be collected by the teacher's PDA through wireless communication.

In case of Self-healing, a student may face problems regarding memory, processor speed, application failure, etc. ETS Self-healing unit of the student's PDA will periodically monitor, as well as save, the status of these issues as a logstatus database file. The screen shot, shown in figure 6, is taken from one student's PDA after running a wireless exam application in it. This database file is maintained in all other devices as well.

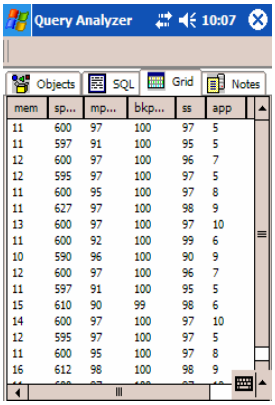

Figure 6. Status of a device

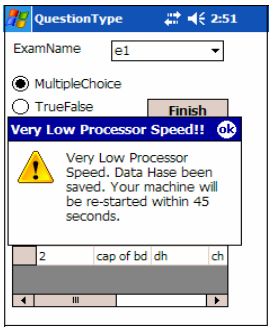

Figure 7. Message to the user regarding low processor speed

By analyzing all these values, another student's PDA identifies the problem of possessor speed since its rate of change over time (the slope) decreases abruptly. After further investigation to verify the problem, ETS Self-healing finds that the device needs to be restarted. Without any delay, ETS saves all other unsaved important information and prompts a message to the user, which is shown in figure 7.

In the second application, named online survey system, a company representative conducts a survey among its customers. Before starting the survey, each

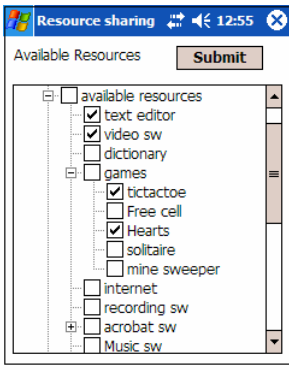

Figure 8. Resource sharing

participant completes a form and submits it to a resource manager (here the PDA of company representative), which indicates the list of resources they are willing to share. Such a form is shown in figure 8. One customer doesn't have a text editor, which is needed to participate in the survey; he gets it from another customer (who wants to share it) via the resource manager.

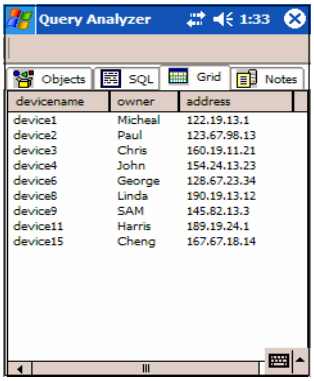

Figure 9. Device discovery

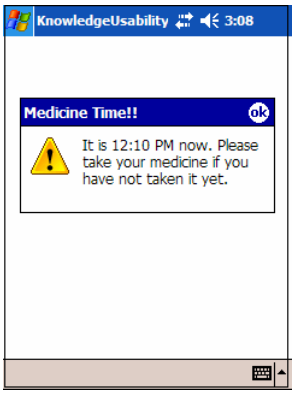

Figure 10. Message about medicine time

To perform resource sharing, first the resource manager needs to discover the devices available within its contour. In the prototype of MARKS, this is simply achieved by each device simply using a broadcasting scheme. After getting all the names and addresses, the resource manager then stores this information along with each particular device name. Figure 9 shows an example of such a database file maintained by the resource manager.

The implementation of Knowledge Usability demands more advanced context awareness (location, situation, time, etc.), which are not currently implemented. Due to this limitation, in our first prototype, only time is considered as a context regarding Knowledge Usability. After checking the current system time, MARKS helps the user with the upcoming actions. Figure 10 shows an interaction scenario of MARKS with a user regarding taking medicine.

For evaluating the performance of MARKS we collected data by running MARKS prototype and some of the designed applications. We analyzed the battery power consumption and network traffic for running MARKS prototype in PDAs. We found that the needed battery power for running MARKS is very negligible.

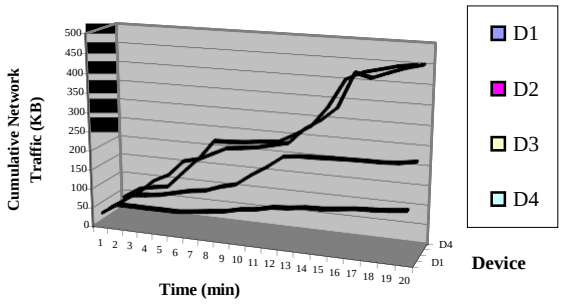

Figure 11. Time (min) vs. cumulative network traffic (kb) for MARKS

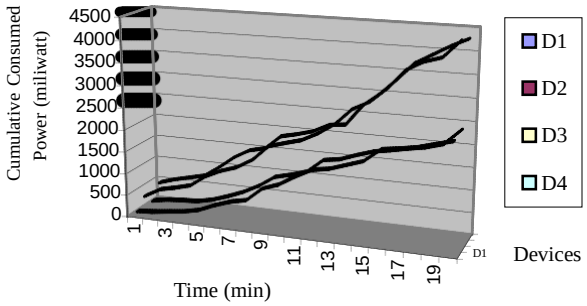

Figure 12. Time (min) vs. cumulative consumed power (miliwatt) for MARKS

Figure 11 and 12 present the network traffic and status of battery power for four devices where MARKS prototype is running. At time 13, device D2 requested a service and device D4 agreed to offer that service. The network traffic increased a lot for both of these devices from this point. The sharp change in network traffic for these two devices is shown in figure 11. Figure 12 presents the battery consumption for the same devices. The sharp change of the status of the battery consumption for D2 and D4 (represented by the upper two lines on the graph) after time 13 is due to the above fact. The message passing between these two devices increased a lot and as a result the consumed power also became high.

4. CURRENT STATE OF THE ART

Even though more researchers are involved in designing middleware solutions for embedded devices running on a pervasive computing environment, no optimum solution has been provided to date. Knowledge Usability, a key aspect to determine an application's behavior, is overlooked. In Reflective middleware [7, 17], the concept of user profile, a crucial segment of Knowledge Usability, was introduced but not utilized to its full capacity. A simple yet powerful algorithm, to unearth available resources, has not yet been devised. Most of the approaches follow the resource announcement policy. Self-healing, a term mainly used in autonomous computing which covers not only fault tolerance but also resource recovery, should also be adapted in mobile computing middleware. Heartbeat tactic is the most prevalent scheme for fault detection, which sometimes can proceed to erroneous direction.

Reconfigurable Context Sensitive Middleware (RCSM+) [3] mainly deals with situation-awareness, ephemeral group management and autonomous coordination for information dissemination. Knowledge Usability, Resource Discovery, and Self-healing issues were not addressed in their design. Gaia [18] tries to solve the problem of ubiquitous computing by introducing a general operating system middleware, which exports and coordinates the resources contained in a physical space. They introduce the idea of active space that converts a physical space and its ubiquitous computing devices into a programmable computing system. Gaia's activities, however, are confined only within the active space. MIT's Oxygen project [19, 20] turns the inactive environment into an empowered one to facilitate the users. This project focuses on new adaptive mobile devices, new embedded distributed computing devices, intelligent knowledge access technology, automation technology etc. At present systems can divert a user in many explicit and implicit ways, which may reduce his/her effectiveness. Project Aura [21] rethinks system design to address this problem. Aura tries to provide each user computing and information services at every level regardless of location. Gaia, Oxygen, and Aura [21, 22] show splendid performance inside the smart space. But their focus is different, as they try to provide all the facilities they mentioned inside a particular smart space. Consequently, these are not the ultimate solution for mobile devices running in a pervasive computing environment.

Other noteworthy middleware for mobile devices includes LIME [8, 23], XMIDDLE [24], TSpaces [25, 26], Mobiware [27], and ALICE [28]. LIME, supporting scarce context-awareness and inadequate ad hoc communication, is the result of a development

process assimilating formal modeling integration and application development. TSpaces provides a common platform to facilitate the linkage of all systems and application services. In this case, the method of communication is flexible. Server software containing data are stored on fixed and powerful machines; this is inappropriate in an ad-hoc communication environment. Xmiddle uses a tree-structure for storing data. Here, unit of replication can be adjusted to accommodate both device and application needs. It is appropriate for mobile computing since it targets ad-hoc networks. However it is implemented using Extensible Markup Language (XML) that increases the communication overhead.

These middleware solutions along with other existing ones did not provide a single answer from the standpoint of Fault Tolerance, Resource Discovery and Knowledge Usability. A comparison table of the middleware discussed above is presented below. Table 1 summaries the features supported and not supported by the popular middleware.

Table 1. Comparison of Middleware features

Middleware	Features Supported	Features Don't Supported
RCSM	Context and Situation-awareness, Ad-hoc Communication, Partial Knowledge usability	Resource Discovery, Self-healing
Gaia	Context and Situation-awareness, Resource Discovery, Partial Self-healing	Needs fixed infrastructure; don't support Ad-hoc Communication
Oxygen	Intelligent knowledge access technology, Context and situation awareness, Resource Discovery	Needs fixed infrastructure; don't support Ad-hoc Communication
LIME	Scarce Context-awareness and inadequate ad-hoc communication	Resource Discovery, Self-healing, Knowledge usability
MARKS	Context-and Situation-awareness, Ad-hoc Communication, Partial Knowledge usability, Resource Discovery, Self-healing	Full fledged Knowledge Usability and Byzantine fault recovery is not supported fully in the current prototype

5. CONCLUSION

In this paper, we have addressed the Knowledge Usability, Resource Discovery and Self-healing properties of pervasive computing. We have incorporated those in MARKS as middleware services. Knowledge Usability is the ability of an application to more accurately help the user to plan his/her daily activity. Knowledge Usability plays a vital role in reducing the burden of administrative and configuration work on a user's behalf. To make an appropriate decision in any particular event, it is crucial to have an idea about the users' preferences and dislikes. . "Knowledge Usability", an attribute of middleware (MARKS), will facilitate the adaptation of the application's behavior depending on a preference and requirements.

Resource Discovery, a fundamental component of any system running in a pervasive computing environment, queries for resources providing a specific service. Since the existing middleware is not capable of offering efficient solutions to the needs of the devices, a new approach is badly needed. Further work needs to be done in the areas of efficiently providing services to many devices at one time, reducing the seeking time, and also tracking the other devices as substitute resource providers. MARKS includes a resource discovery unit that guarantees the best service discovery mechanism.

Fault detection and resource recovery are covered in the Self-healing unit. Sensible fault prediction, along with dynamic resource recovery, make MARKS' self-healing unit unique. MARKS ensures the least possible time not only to detect any kind of fault but also to restore its smooth functionality.

In our first prototype, we implemented the entire Resource Discovery, device discovery, ETS Self-healing, and a preliminary version of Knowledge Usability unit of MARKS. A simple yet effective mechanism for maintaining security and privacy are also incorporated.

As we are working on MARKS, we are trying to make it more robust. In our Self-healing unit, our future goal is to incorporate facilities to recover Byzantine faults, as these kinds of faults are very difficult to recover. We are also planning to design benchmarks that will allow us to not only analyze the performance of the entire middleware, but also the performance improvement factor for each unit in the middleware.

MARKS can have a very positive effect on the hectic and busy life styles of people today. MARKS may ease everyday life in a transparent manner. It can act as an effective virtual assistant for professionals. We have already developed it called Ubicomp Assistant [30] on top of MARKS. We have also developed Healthacare Aide[32] for doctors using MARKS middleware. Hence, MARKS has already showed its power, usability and strength to improve pervasive computing. As a long-term benefit, MARKS will enhance the existing market of embedded devices. It will also facilitate the massive technology transfer in both industry and academia.

REFERENCES

[1] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing", *ACM Symposium on Principles of Distributed Computing, 1995 (PODC'95 invited lecture)*, 1995, URL: http://www-2.cs.cmu.edu/afs/cs/project/coda/Web/docdir/podc95.pdf (accessed in May 2005)

[2] G. Abowd, and E.D Mynatt, "Charting Past, Present, and Future Research in Ubiquitous Computing", *ACM Trans. Computer Human Interaction*, vol. 7(1), March 2000, pp. 29-58, URL: http://www.cc.gatech.edu/fce/pubs/tochi-millenium.pdf (accessed in May 2005)

[3] 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: http://www.eng.auburn.edu/users/yuwang/Compsac02-pres.pdf (accessed in May 2005)

[4] Ubiquitous Computing, URL: http://www.ubiq.com/hypertext/weiser/UbiHome.html (accessed in May 2005)

[5] 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*, CSREA Press, Las Vegas, NV, USA, June 2005, pp. 206-209.

[6] T. Kindberg, and A. Fox, "System Software for Ubiquitous Computing", *IEEE Pervasive Computing*, January-March, 2002, pp. 70-81, URL: http://www.champignon.net/TimKindberg/kindberg-fox-ieeepvc.pdf (accessed in May 2005)

[7] L. Capra, W. Emmerich, and C. Mascolo, "Reflective Middleware Solutions for Context-Aware Applications", *Proceedings of Reflection 2001, Kyoto, Japan, volume 2192 of LNCS, Springer Verleg.* September 2001, pp. 126-133.

[8] A. L. Murphy, G. P. Picco, and G. C. Roman, "Lime: A Middleware for Physical and Logical Mobility", *Proceedings of the 21st International Conference on Distributed Computing Systems (ICDCS-21)*, May2001, URL: http:// www.cs.rochester.edu/ u/ murphy/ papers/ icdcs01.pdf (accessed in May 2005)

[9] M. Sharmin, S. Ahmed, and S. I. Ahamed, "SAFE-RD (Secure, Adaptive, Fault Tolerant, and Efficient Resource Discovery) in Pervasive Computing Environments ", *IEEE international Conference on Information Technology (ITCC 2005)*, Las Vegas, NV, USA, April 4-6, 2005.pp. 271-276.

[10] D. Garlan, V. Poladian, B. Schmerl, J. P. Sousa, "Task-based Self-adaptation" , *Proceedings of the ACM SIGSOFT 2004 Workshop on Self-Managing Systems (WOSS'04)*, Newport Beach, CA, Oct/Nov 2004, URL: http://www-2.cs.cmu.edu/~able/publications/WOSS04/ (accessed in May 2005)

[11] C. Dabrowski, and K. Mills, "Understanding Self-healing in Service-Discovery Systems", *Proceedings of the first workshop on Self-healing systems***,** Charleston, South Carolina, 2002 pp. 15 – 20,, ISBN:1-58113-609-9, URL: http://www.antd.nist.gov/~mills/papers/WOSS2002-09-30-b.pdf (accessed in May 2005)

[12] G. Banavar, J. Beck, E. Gluzberg, J. Munson, J.B. Sussman, D. Zukowski, "Challenges: an application model for pervasive computing", in *Mobile Computing and Networking*, 2000, pp. 266-274

[13] S. Ahmed, M. Sharmin, and S. I. Ahamed, ETS (Efficient, Transparent, and Secured) Self-healing Service for Pervasive Computing Applications, to appear in Journal of Network Security

[14] Secret Sharing, URL: www.cmpe.boun.edu.tr/ courses/ cmpe471/ spring2003/ download/cmpe47109-2003.ppt (accessed in May 2005)

[15] F. Zhu, M. Mutka, and L. Ni, "*Classification of Service Discovery in Pervasive Computing Environments*", URL: http://www.cse.msu.edu/~zhufeng/ServiceDiscoverySurvey.pdf (accessed in May 2005)

[16] 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 2005)

[17] Reflective Middleware, URL: http://dsonline.computer. org /middleware/RM.htm (accessed in May 2005)

[18] R. Cerqueira, C. K. Hess, M. Roman, and R. H. Campbell, "Gaia: A Development Infrastructure for Active Spaces", *In Workshop on Application Models and Programming Tools for Ubiquitous Computing (held in conjunction with the UBICOMP 2001)*, Sept. 2001, URL: http://www.tecgraf.pucrio.br/publications/artigo_2001_gaia.pdf (accessed in May 2005)

[19] M. Dertouzos, "The Oxygen Project," *Scientific American*, Vol. 281 No. 2, August 1999, pp 52-63.

[20] The Oxygen Project, URL: http://oxygen.lcs.mit.edu/ Overview.html (accessed in May 2005)

[21] Project Aura,URL : http://www-2.cs.cmu.edu/~aura/ (accessed in May 2005)

[22] J. P. Sousa, and D. Garlan "Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments", *Software Architecture: System Design, Development, and Maintenance (Proceedings of the 3rd Working IEEE/IFIP Conference on Software Architecture)*, August 25-31, 2002. pp. 29-43.

[23] LIME, URL:http://www.cs.rochester.edu/u/murphy/ 4.pdf (accessed in May 2005)

[24] C. Mascolo, L. Capra, S. Zachariadis, W. Emmerich, "XMIDDLE: A Data-Sharing Middleware for Mobile Computing", *J. Wireless Personal Comm.*, vol. 21(1), Apr. 2002, pp. 77-103, URL : http://www.cs.ucl.ac.uk/staff/ s.zachariad/ papers/ mw.pdf (accessed in May 2005)

[25] P. Wyckoff, S. W. McLaughry, T. J. Lehman, and D. A. Ford, "T Spaces", *IBM Systems Journal*, 37(3): 454–474,1998,

URL : http://www.research.ibm.com/ journal/ sj/ 373/ wyckoff.html (accessed in May 2005)

[26] TSpaces, URL : http://www.cs.berkeley.edu/ ~ravenben /research/tuplespace/tuplespace.PPT (accessed in May 2005)

[27] A. T. Campbell, "Mobiware: QOS-aware middleware for mobile multimedia communications", *Proceedings of the IFIP TC6 seventh international conference on High performance networking VII*, White Plains, New York, United States, 1997, pp: 166 – 183.

[28] M. Eichberg, and M. Mezini, "Alice: Modularization of Middleware using Aspect-Oriented Programming", *Software Engineering and Middleware (SEM) 2004*, 20-21 September 2004, Linz, Austria, URL: http://www.st.informatik.tu-darmstadt.de/database/publications/data/Alice.pdf?id=103 (accessed in May 2005)

[29] Sheikh I. Ahamed, Moushumi Sharmin, Shameem Ahmed, Michael J. Havice, and Suresh Anamanamuri, An Assessment Tool for Out of Class Learning using Pervasive Computing Technologies, *Journal of Information, vol 8(5)*, September 2005.

[30] Moushumi Sharmin, Shameem Ahmed, and Sheikh I. Ahamed, "Ubicomp Assistant: An Omnipresent Customizable Service using MARKS, " *To appear in ACM SAC 2006*, Dijon , France, April, 2006.

[31] Moushumi Sharmin, Shameem Ahmed, and Sheikh I. Ahamed, "An Adaptive Lightweight Trust Reliant Secure Resource Discovery for Pervasive Computing Environments, " *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computer and Communications (PerCom 2006)*, Pisa, Italy, Mar 2006, pp. 258-263.

[32] Moushumi Sharmin, Shameem Ahmed, Sheikh I. Ahamed, Munirul Haque, and Ahmed J Khan, "Healthcare Aide: Towards a Virtual Assistant for Doctors, Patients, Nurses and Resident Doctors Using Pervasive Middleware, " *Proceedings of the 1st Workshop on Ubiquitous and Pervasive Health Care (UbiCare 2006) in conjunction with Fourth Annual IEEE International Conference on Pervasive Computer and Communications (PerCom 2006)*, Pisa, Italy, Mar 2006, pp. 490-495.

[33]http://www.epaynews.com/statistics/mcommstats.html#7