# A Framework for enabling Service migration in Ubiquitous Computing

Irene Antony Tharayil and Dr. Rajasree M. S.

Masters Student, Department of Computer Science and Engineering, College of Engineering Trivandrum

Professor, Department of Computer Science and Engineering, College of Engineering Trivandrum
ireneantony88@gmail.com, rajasree40@gmail.com

## ABSTRACT

*Ubiquitous Computing is a computing paradigm in which applications move with the user. This requires applications to migrate automatically to the environment the user is present and adapt to different contexts and resource availabilities. An important requirement of a ubiquitous computing system is this automatic adaptation of applications to the new environment. At the time of migration, there could be incomplete services that are still in a running state. Such services may have to be suspended and transferred to destination before they are completed. In this paper, we propose a service migration framework for migrating partially executed services. When multiple destinations are available for the migration, an Ontology Server selects one after evaluating a variety of attributes such as energy level, computational ability, availability etc. Migrations are handled at thread level for accuracy. In particular, migration of response-awaiting-services is handled based on the communication behavior of the service. These services are divided into two categories viz. Continuous-Interaction-Demanding Services and Limited Interaction Services based on the nature of their interaction with the user. Two distinct protocols are then proposed for handling the migration of these services.*

## Categories and Subject Descriptors

*D.3.3 [Programming Languages]: Language Constructs and Features – abstract data types, polymorphism, control structures. This is just an example, please use the correct category and subject descriptors for your submission. The ACM Computing Classification Scheme: http://www.acm.org/class/1998/*

## General Terms

*Your general terms must be any of the following 16 designated terms: Algorithms, Management, Measurement, Documentation, Performance, Design, Economics, Reliability, Experimentation, Security, Human Factors, Standardization, Languages, Theory, Legal Aspects, Verification.*

## Keywords

*Keywords are your own designated keywords.*

## 1. INTRODUCTION

Ubiquitous computing is future generation model of human-computer-interaction in which all functions are invisibly integrated into a surrounding system without user awareness. A ubiquitous network includes not only desktop computers, laptop computers and mobile phones but also various ordinary objects such as furniture, clothes, etc. These devices enable users to access the required services from anywhere.

### 1.1 Service Migration

Ubiquitous computing environments may have different devices, services, applications, architectures etc. The problem of adapting to changes in resource availabilities, the movement of user to and from various environments and the failure of devices are important issues in a Ubiquitous computing environment. Suppose a person migrates from one system to another while he is running an application. Then the Ubicomp application should migrate its services to the new environment to provide uninterrupted access to the user. Service migration is the process in which a service is migrated from a device A to another device B that supports the particular application. It needs to make use of whichever resources are available in the environments that allow users to carry on with their tasks without interruption.

### 1.2 Migration of partially executed services

In real scenario, migration is not guaranteed to occur at the end of a function/method or after the completion of a service. There may be need of migrating services in the middle of their execution. Suppose a user U moves from the environment of one system, say A, to that of another, say B, while executing a service X. After sensing the movement, service X is to be stopped in environment A and is to be resumed from that particular state in environment B. In this paper, we propose an architectural framework for handling such a service migration.

### 1.3 Migration of Response-awaiting-services

Different services provided by a computing system can be of very different characteristics. Migration of partially executed services needs some special care particularly in the case of response-awaiting-communication-services. Response-awaiting-communication services are those services that take part in prolonged interactive communication sessions with the user. It may be a long database transaction, a multi-player game etc. A set of application level protocols for the migration of such services, which take into account their communication patterns, is proposed in this paper. A prototype of the proposal is also created.

## 2. LITERATURE REVIEW

Initially, service migrations focused solely on replicating the service in one node and then transferring to the target, deleting it from the original. The migration transaction proposed in [1] comprises of three steps: (1) Replicate S3, including its state, from node 1 to node 2 (i.e. copy it physically). (2) Delete S3 on node 1. (3) Update the mapping on the service directory, so that nodes using this service can find its new location. In [2], migration is made possible with the help of mobile agents. Two mobile agents named Domain Mobile Agent (DMA) and Main Mobile Agent (MMA) play an important role in the migration process. When a user wants to perform a service, he requests for a code from a Domain Code Server (DCS). If the code is not already present in the Local Code Buffer (LCB), it requests for the code from the Main Code Server (MCS). Once the code is fetched, the MCS transfers it to the DCS and then to the LCB. This requires MCS to have runtime codes available with it, which may not always be possible. A framework that uses a context aware service migration mechanism is discussed in [3]. The implementation uses OSGi (Open System Gateway Interface) framework to implement service migration and service diffusion.

### 2.1 Migration using a universal language

The issue of what to migrate and what not to migrate is discussed in [4]. The work divides the entire system into two: included state and excluded state. The former includes the environment itself, its entire nested environment and its state of computations. The latter includes computations that are isolated from the environment. The decision is made with the help of a checkpointing mechanism. By traversing all objects reachable from a set of well-defined roots, the checkpointing mechanism captures the state of the computations. The automatically captured checkpoint is restored at the receiver side. In all the above cases, state of the system is migrated assuming that all the applications are present in the system. The similar environments may be identified using semantic ontology. One such method is used in [5] for which migration needs a repository of keywords. Here, similarity is identified by first converting languages to a common format using a Context Computing Language (CCL). Programming language keywords are converted to CCL with the help of a CCL repository.

### 2.2 Service migration architectures

Some high level architecture had been proposed for handling heterogeneity in the event of service migrations. One such system is Aura [6]. The Aura Project is based on the concept of personal Aura, which acts as a proxy for the mobile user it represents. When a user enters a new environment, its Aura marshals appropriate resources to support the user's task. Furthermore, an Aura captures the constraints that the physical context around the user imposes on the tasks requiring several information sources and applications. To enable the action of a personal Aura, an architectural framework that clarifies which new features and interfaces are required at system and application levels is used in [7]. The framework also captures the nature of the user's tasks, personal preferences, and intentions.

Many researches are going on in this area, and as a result, a new middleware architecture known as GAIA has taken form. GAIA converts physical spaces and ubiquitous computing devices they contain into a programmable computing system. It is similar to traditional operating systems in that it manages the tasks common to all applications[8]. GAIA provides core services including events, entity presence (devices, users and services) discovery and naming [9].

By specifying well-defined interfaces to the services, applications may be built in a generic way so that they can be run in arbitrary active spaces. GAIA uses CORBA and consists of a number of different types of agents performing different tasks such as discovery, context sensing, event distribution etc. [10]. Each user also has an agent that keeps personal information and acts as a proxy in a variety of settings. There are also application agents that help users perform various kinds of tasks such as playing music and drawing. A new framework was proposed in [11] with the aim of maximizing usage of resources available and minimizing user distraction. But GAIA is the most popular one and many researches are going on considering GAIA as the base framework.

## 2.3 Architectures that support application polymorphism

In [12] the concept of application polymorphism and a method to handle different types of polymorphism is introduced. Here, a trigger from user is required for migration. When an application is suspended, application state and structure is stored in file and it is transferred. Service queries an Ontology Server and from there semantically similar components are identified. Another work [13] enables an application to modify its structure to adapt to the resources in a ubiquitous computing environment. The application framework allows an application to be decomposed into smaller components, each of which can be independently adapted and recomposed to obtain a semantically similar application.

The adaptation mechanism uses ontologies and context information to choose appropriate components and devices to support the application. Many architectures are put forward to enable seamless user experience when the user moves around. For this, certain adaptations to deal with heterogeneity are needed. First is type adaptation to check and identify the semantic similarity among applications. Second is device discovery which is to identify on what device a particular service may be performed. Third is cardinality adaptation, which uses context as well as user preferences to choose the best device to run, if multiple devices are present. Among these, the first two may be done with the aid of ontology servers.

## 3. MOTIVATIONAL FACTORS

In all the systems discussed so far, migration is assumed to occur at the end of a function. But in a real life scenario, migration is not guaranteed to occur after the completion of a function or method. There may arise a case where a partially executed service is to be migrated. None of the existing systems address this issue.

The concept of thread migration that is used in virtual machines could be adapted and applied to Ubicomp systems to ensure the correct migration of partially executed services.

This has been the prime motivational factor behind this research. If a user centric framework could be developed, it could make migration a lot easier. This was a motivation to develop a framework, which can be used by a developer for creating his own application, so that user migration will be made easy and invisible without being concerned about low level details.
Ubicomp devices may be distributed in an environment and they will be working on batteries. These batteries may need to be recharged periodically. As the user moves, ‘migration’ demands moving of services to a device nearest to the user. However, the energy level of the target system is to be considered before doing the migration. If it is lower than a minimum energy level, migration must not be done to that system. Such an energy level consideration is present in Adaptive Wireless Sensor Network [1]. This was motivation for introducing an energy level parameter to the migration.

Another consideration is regarding services that require high computational capability. A ubiquitous environment will have devices with different computational capabilities, ranging from small PDAs to highly sophisticated high performance computers. The presence of such lightweight devices in the environment should be acknowledged by a before service migration. High computational services should not be migrated to devices with a low capability. No such consideration is present in existing systems.

## 4. PROPOSED SYSTEM

The proposed system aims to achieve the following:
Handles services in a machine
Selects a proper destination device to migrate
Migrate services including partially executed services.

Handles Response-awaiting-services (those services that are waiting for a reply from the user or from some other service).

A program or a group of program wrapped up to perform a single job/task.A service consists of one or more threads. The thread level migration can be performed with the aid of jikesRVM [14],Padmig[15] etc. By using these as the base layer, thread migration can be implemented in a ubiquitous computing environment Each service is developed as child process of proposed framework.

System consists of meta-information about each service. This meta-information consists of Service Name, Service Id, Related Files, Related libraries, Communication Protocol, Class containing Main Function. User moves out from the vicinity of machine A i.e. it moves from environment A. A then starts monitoring the ontology server (will be discussed later) to identify the location of user. On finding the location it searches for available environments. Then it selects an environment satisfying the selection parameters.

The selection parameters includes:
Availability of related libraries, services etc. It checks initially for the availability of related libraries.

Computational Capability: High computational services should not be migrated to devices with low capability. If no high capability device is found, result alone is displayed in light weight device found.
Distance with user : To find closest machine to user
Energy Levels: It searches for a machine with high energy levels.
The main objective of this work is to develop a system for response-awaiting-service migration in Ubiquitous Computing.It is explained in following subsection.

### 4.1 Proposed response-awaiting-services' migration

In this paper, we propose a set of application level protocols for migrating services that are awaiting a response from the user (called "response-awaiting-services"). The need for multiple protocols is necessitated by a fundamental difference in the nature of these services. Some services need to interact with the user continuously over a long period of time to complete successfully. Some services, on the other hand, may not require such intensive interaction with the user. Instead, the processing is done for a long time before sending a response back to the user.

Based on the above discussions, communicating services may be classified into two:
Type I Services: Continuous-Interaction-Demanding Services
Type II Services: Limited Interaction Services

#### *4.1.1 Continuous Interaction Demanding Services' Migration (CISM)*

Type I Services or Continuous-Interaction-Demanding Services are those services, which need frequent interaction with the user. Because of this very nature, the protocol used for migration of such services should make sure that no messages are lost during the migration process. The proposed protocol for Continuous-Interaction-Demanding Services works much like the Mobile Internet Protocol. They are present in Fast Response system.

Suppose a user moves from the environment of device A to that of device B, after initiating a database transaction. First response to the transaction is sent to A from the central database server. Knowing that user has already moved to the environment of B, A sends the details about the migration to the central database and also forwards the received response to B. Now that, the central database has came to know about the new environment of the process, any future responses will be sent directly to B and not to A.

The above steps may be summarized as follows:

Initially, as the user is in A's environment, responses to the user are sent to A.
User moves to a new location. Server is unaware of this.It sends the next response to A itself.
After finding the new location, A forwards the response message to it. B's environment details are send to server.

All the remaining messages from the server are sent directly to the new destination B until a new "migrate" command is initiated.

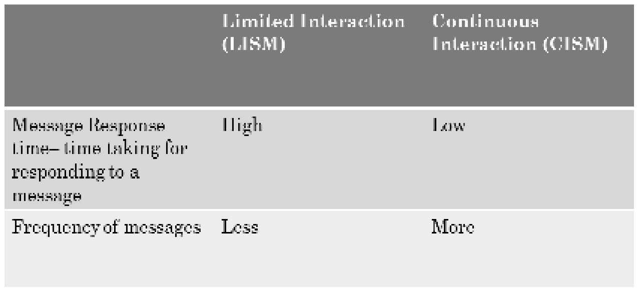
##### *4.1.1.1 Limited Interaction Services' Migration (LISM)*

Type II Services or Limited Interaction Services are services that take a long processing time before sending a response to the user. The framework works by identifying such services in the initial setup phase itself. Suppose a user moves from the environment of system A to that of B while performing a task. The information that the user has moved is sent to the centralized database immediately after sensing the movement. Finally, the result after the long computation is sent directly to B.

The above steps may be summarized as follows:
Initially, as the user is in A's environment. Responses to the user are sent to A.
User moves to a new environment B.A notifies the server that the user has moved out of A's environment. It also sends the new address of service, which is of B.

The server completes its computation and sends the result and any following responses directly to B.

As results are obtained after long computations, it is assumed that they are sent to the new location after service migration. But if at all some responses are ready before migration, it is held back until migration is completed. Comparison is given in Figure 1.



| | Limited Interaction (LISM) | Continuous Interaction (CISM) |
|---|---|---|
| Message Response time– time taking for responding to a message | High | Low |
| Frequency of messages | Less | More |

Figure 1 Comparison of two migration models

## 5. EXPERIMENTAL SETUP

A prototype for service migration framework was implemented. All services are supposed to have service name, service type, IP, etc as their attributes. We used Intel i3 2.43 GHz with 3GB RAM.

The architectural design is shown in Figure 2. The various components in the system are explained below.

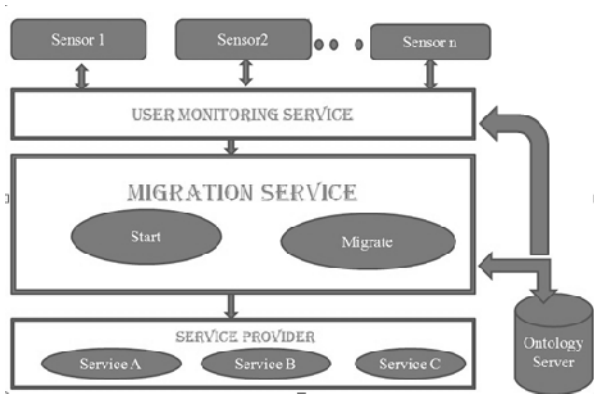


Figure 2 Architectural Design

### 5.1 Sensors

Sensors are various electronic devices placed in the environment to monitor user movement. It can be cameras, IR sensor etc.

## 5.2 User Monitoring Service

The User Monitoring Service monitors the presence of the user in a particular environment. Tracking of user movement and environment identification is done in this module. User may be in vicinity of multiple environments simultaneously. The term “environment” refers to the range of a particular device. If a user has access to a device, then it is said that the user is in the environment of that device. This is supposed to be simulated for this thesis by generating random outputs as the focus of our work is not on identifying the location of the user but instead on the migration of services.

All these processes communicate with each other to know the current location of a user and updates the Ontology Server. It also senses when the user leave- this information is fed to migration service as well as Ontology Server.

## 5.3 Migration Service

The Migration Service is the module where all functionalities related to service migration are implemented. It is the thread, which monitors all the services and coordinates their migration The Migration Service is also preloaded (where it is mostly a static module). It will preload metadata about a service during the loading time itself. When the Migration Service is instantiated with an object of a particular service, it will load information about the service from the Ontology Server. The Migration Service provides functionalities like starting and migrating a service If a user wants to start a service, he will give request through an interface to "start" that particular service. The module then calls the Service Provider to start the corresponding service.

For migrating a service, the service looks up the Ontology Server in order to find out the destination. The destination to migrate is determined based on selection parameters like availability of libraries, remaining energy etc. It requests the Ontology Server for the details of the identified machine.

The ‘migrate’ request provided by the User Monitoring Service will contain information about devices having the user in its environment. The information is returned back to the Migration Service, and it selects the appropriate machine after considering all the parameters. Once the destination is fixed, it will ask the Service Provider to suspend the current service and then initiate a migration by communicating with the Migration Service at the destination. If there are no machines available with enough computational power, then the service is allowed to complete its execution, only after which the service is migrated with the results of the computation.

The migration is done with the help of thread serialization. The suspended thread is first serialized and sent from one system to the Migration Service of the destination system. On receiving the thread, the destination will deserialize the thread and resume its execution. This process is called thread restoration. This thread level migration is performed for better accuracy. PadMig[7] (The Paderborn Thread Migration and Checkpointing) Library is integrated in this module for enabling effective thread level migration. The major issue with service migration in an ubiquitous environment is the migration of partially executed services. It can also be integrating PadMig to our framework.

### 5.4 Service Provider

This module provides various support services to the Migration Service as well as user programs. The Service Provider holds a variety of information about each service. It acts as a parent process or host process to different user services and invokes different services as its child services. It will also monitor energy levels of system. When it drops below the threshold level, it will update the information in the Ontology Server.

All services are supposed to have serviceName, serviceID, sysIP,sysName, portNo etc as their attributes. If migrate request is obtained, certain attributes are changed automatically and the service is migrated. Service provider Handles all the communication. It is implemented with communication protocols specified in CISM and LISM. It takes meta information about service for providing a suitable environment for migration. This meta information is present in Ontology server.

### 5.5 Ontology server

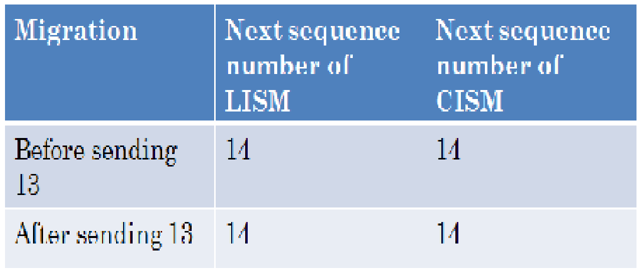
The Ontology Server is a small semantic database, which consists of meta-information about devices. It includes various services supported by the device, its related libraries, the required specification for providing each service etc. It also contains details about type of service as Type I or Type II etc.

The details about devices will have information like specification of the device, the libraries present in the device, the platform the device is running on, Computational capability, Location of device etc. The Ontology Server is a centralized system placed in a safe location and is not mobile. The Ontology Server also receives updates about the energy level of devices. This information is provided as a binary state: low energy or high energy. The distinction between low energy and high energy is made based on a threshold level, which is fixed initially. Anything below the threshold level is considered as a low energy state. The threshold level is published initially and hence known to all devices. When the energy of a device goes below this threshold, the Service Provider (which is discussed in next section) is informed which communicates the event to the Ontology Server. This dynamic update keeps the energy level information of devices in the server database up to date.

## 6. EXPERIMENT

The proposed framework is implemented and tested. All the sensor details generated randomly for this work. A sample application is developed in this framework and tested. For that a client-server based application is developed, which continuously sends and receives some random message which has attached sequence numbers. Client start with a message of sequence number 1 and Server responds with message 2.

This sequence number is given to identify whether the message is lost and also if the message delivery is in correct order. It terminates at 100.In between at some stage a migrate request is generated randomly. State capture is done, the current thread is suspended and migration is performed. Analysis of the migration is depicted in Figure 6-1.



| Migration | Next sequence number of LISM | Next sequence number of CISM |
|---|---|---|
| Before sending 13 | 14 | 14 |
| After sending 13 | 14 | 14 |

Figure 3 Analysis of the result

It is inferred that no loss of messages in the case of LISM and CISM. This type of message handling of response-awaiting-service is not present in any of the systems developed for Ubiquitous Computing Environment.

## 7. CONCLUSION AND FUTURE WORK

In this paper, we propose a user-centric migration framework for performing service migration, which enhances ubiquity and can act as a promising stepping-stone for the future of ubiquitous computing. The detailed design of the framework that supports migration of partially executed services is discussed in this work and a small prototype is also developed. To best of our knowledge this is the first approach in service migration. A set of application level protocols are also developed for services based on their communication behavior. The fully developed system can make service migration in a ubiquitous system easy, transparent and painless and will provide a seamless computing experience to end-users.

As part of future work, we are concentrating on redevelopment of this framework in android environment. Android is more suitable for a mobile environment. Security considerations are also to be integrated in the system. There are many researches going on in this area of security of Ubicomp environment. As part of future work, some mechanism for authentication and message integrity are to be incorporated in the system.

## 8. REFERENCES

[1] Christoph Reinke, Nils Hoeller, Volker Linnemann, "Adaptive Atomic Transaction Support for Service Migration in Wireless Sensor Networks", *Wireless And Optical Communications Network (WOCN), Seventh International Conference OnDigital Object Identifier*, 2010.

[2] Jun Lu, "The Mobile Agent Based Service Migration Mechanism in Wide Area Pervasive Computing System", *WASE International Conference on Information Engineering, IEEE computer society, 2009*.

[3] Davy Preuveneers and Yolande Berbers, "Context-driven migration and diffusion of pervasive services on the OSGi framework", *Int. J. Autonomous and Adaptive Communications Systems, vol. 3, 2010*.

*[4]* Robert Grimm, Janet Davis, Eric Lemar, and Brian Bershad, "Migration for Pervasive Applications", *University of Washington , 2010.*

[5] Shaji R.S,Rajesh R.S, Ramakrishnan B., Rajesh F, "A novel Routing and service migration scheme for communication among heterogeneous devices in pervasive environment, Computer, Communication

and Electrical Technology (ICCCET)", *International Conference on Digital Object Identifier , Mar. 2011*

*[6]* Joo Pedro Sousa and David Garlan Aura: "An Architectural Framework for User Mobility in Ubiquitous Computing Environments*", Bosch, Gentleman, Hofmeister, Kuusela (Eds), Kluwer Academic Publishers, pp.29-43, Aug. 2002.*

[7] David Garlan, Daniel P. Siewiorek, Asim Smailagic, and Peter Steenkiste, "Project Aura: Toward Distraction-Free Pervasive computing, *IEEE pervasive Computing, special issue on "Integrated Pervasive Computing Environments", Vol. 21, No. 2, April-June, 2002*.

[8] Manuel Roman , Roy H, "Gaia: Enabling active spaces", *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system ,September 2000*.

[9] James Bresler Jalal Al-Muhtadi Roy Campbell, "Gaia Mobility: Extending Active Space Boundaries to Everyday Devices", *Proceedings. 24th International Conference IEEE on Distributed Computing Systems Workshops, 2004*.

[10] Manuel Romean, Christopher Hess, Renato Cerqueira, Anand Ranganat, Roy H. Campbell, Klara Nahrsted "Gaia: A Middleware Infrastructure to Enable Active Spaces" *IEEE, Software Technologies for Future Embedded and Ubiquitous Systems, 2004*.

*[11]* Ranganathan A, Chetan S, Campbell R."Mobile polymorphic applications in ubiquitous computing environments" *IEEE, Mobile and Ubiquitous Systems: Networking and Services, 2004.*

[12] Khin Phyo Thant, Thinn Thu Naing"A Migration Framework for Ubiquitous Computing Applied in Mobile Application"*Information and Telecommunication Technologies, 6th Asia Specific Symposium, 2005.*

[13] Khin Phyo Thant, Thinn Thu Naing "A Migration Framework for Ubiquitous Computing Applied in Mobile Application" Information and Telecommunication Technologies, 6th Asia Specific Symposium, 2005.

[14] Raffale Quitadamo Giacomo Cabri Letzia Leonardi "Mobile Jikes RVM : A framework to support transparent Java thread migration" *Science of Computer Programming vol 70 Issue 2-3 Feb 2008.*

[15] PadMig, http://padmig.sourceforge.net/