

Architecture for Service Mobility across Multiple Provider Domains

M. Jaseemuddin¹, R. Khuwaja¹, B. Major¹, B. Nguyen¹, M. Loryman², and P. Buckle²

¹ Computing Technology Lab, Nortel Networks, 3500 Carling Ave, Nepean, ON K2H 8E9, Canada

² Advanced IP Services and Management, Nortel Networks, London Road, Harlow, Essex, UK CM17 9NA

Abstract

The concept of Virtual Home Environment (VHE) is to present to UMTS users the same personalized features, user interfaces and services in any network and terminal irrespective of where the user is located. This is a multi-dimensional problem that deals with the issues of service mobility, user-profile mobility, signaling (service level agreements), and some service specific problems, such as information consistency etc. The ACTS project CAMELEON is concerned with the development of agent based services to validate the concept of the Virtual Home Environment (VHE). One of them is the Virtual Address Book (VAB) service. The focus of our work for CAMELEON consortium is to develop a VAB service using a service architecture that meets

solution for information consistency in that context.

1. Introduction

Providing personal mobility with the ability of accessing personalized multimedia services anywhere anytime is the goal of the next generation wireless system. The concept of Virtual Home Environment (VHE) is to present to UMTS users the same personalized features, user interfaces and services in any network and terminal irrespective of where the user is located [12]. In addition to voice, some wireless service providers have also started offering data services. But, these service offerings are far short of the goal set by VHE. Supporting full personal mobility across multiple networks, run and managed by different network operators, involves technical challenges and requires a more cooperative business model. This is a multi-dimensional problem that deals with the issues of service mobility, user-profile mobility, signaling (service level agreements), and some service specific issues such as information consistency. In this paper we will focus on the aspects of service mobility.

When a person moves from one provider network to another, he may observe a number of changes, which may occur in the terminals he was using, in the services he was getting, and in the presentation of the services he was

network boundary. Although the transition need not be seamless, the user must be able to access his set of subscribed services in the personalized form through any terminal. The service presentation is only constrained by the network support and the terminal capability. The user is notified of the constraints if the services cannot be accessed in the personalized fashion.

In this paper we present a model of service mobility and focus on the service architecture to make a service mobile. This model is aimed at providing VHE. In this model, when a user roaming in a foreign network wants to access his subscribed service that is not offered by the network operator, a new instance of the service can be dynamically moved there. The operator of the serving network gets the new instance from the service provider offering that service. This model of service mobility offers several additional benefits. It allows rapid deployment of new services in the network. This may be in the interest of the network operator, because it can publish subscriptions for the service to attract new customers. It also gives service providers opportunities to explore new business models. For example, a provider can lease certain services to other network operators, which can be allowed to offer the same service to its customers. Thus, the service user-base expands relative to the popularity of the service, which can be

For services developed using this architecture, the actual service components can be moved to a serving network. The availability of enabling technologies, such as Java, CORBA and mobile agents, simplifies the implementation of services using this architecture. The service architecture also supports a relatively simple model of accessing the service from a remote location. It is flexible in accommodating services with diverse resource requirements by separating their functional and mobility aspects. We demonstrate the applicability of this architecture by developing a *Virtual Address Book* (VAB) service [1].

The paper is organized as follows. In section 2, different approaches to service mobility are discussed, and a service mobility protocol is presented to make the service mobile across provider domains. Section 3 presents a mobile-service architecture. In section 4, the design of virtual address book is described. The VAB is a personal mobile service developed using the service mobility architecture described in section 3. The paper is concluded in section 5.

2. Service Mobility

This section sets the stage for presenting the mobile service architecture, which is described in the next section. Section 2.1 explains the service environment and

negotiation between the service provider and the network operator to move a mobile instance of the service to the serving network.

2.1 Service Environment

A *network operator* provides its users wireless access to some services, such as voice phone, data service, mobility portal etc. Examples of data service include internet connection for web surfing, email, file transfer etc. The network can host multiple services. The computation environment of a network operator typically comprises of a number of servers running some infrastructure software, such as OS, service platforms etc, and some services. A service may run on a single server, or may be distributed across multiple servers. The *service provider* for a service is the provider that owns and offers that service. Some service providers may provide value-added services. Network operators may also provide some services. In this paper, we assume that network operators are also service providers for some services. The *home network* of a user is the network that manages the user profile. A user may subscribe services from several providers, but all those services are accessible through his home network. When a roaming user connects to a foreign network and accesses his subscribed services, then the foreign network is called the *serving network*.

terminals vary in their capabilities and features. Some terminals are relatively less mobile than others. A user may use *any kind of terminal* to access his subscribed services through *any network*.

2.2 A Simple Service Architecture

Services are composed of many components. Most of the components are service specific, but two components are required by the service architecture for every service, as discussed in this section. First component is a *service factory* that is responsible for creating and assembling all the components of the service. It is also capable of maintaining a repository of service instances of that service type. A service provider has one service factory for each service type it offers. Another component is a service handle that performs two main functions. One, it provides a service interface to the outside world. Two, it also connects all the components of the service. When one service component wants to interact with another, it gets the component through the handle.

2.3 Approaches to Service Mobility

Services are mobile when they are accessible through different network domains. There are three main approaches to make a service mobile. These approaches are not mutually

serving network operator establishes a communication channel between the service access point in its domain and the service provider domain. The service access point is typically the user terminal connected to the serving network. The service data moves across the channel using some form of request-response protocol or RPC. The terminal must be capable of receiving and presenting the service data. If there are multiple users accessing the same service simultaneously, the network operator must support each remote connection to prevent denial of service.

2. Deploying a service proxy

Another way of accessing a foreign service is to deploy a service proxy in the serving network. The service proxy typically serves a single user. It can be implemented as a mobile agent that mediates the requests between the terminal and the service running in the service provider domain. It may have limited storage capacity for caching service data. It simplifies the terminal software design by managing the connections to the service and hiding the details of service access, and in some cases even by massaging the service data. This can be implemented using the Client/Server/Agent model of mobile computing [6].

3. Cloning and moving the service components

each service may have its own strategy for the service mobility.

In this approach mobility is a feature of the service design. Which components are moved to the serving network depends upon several factors. Some are internal factors related to the nature of the service and its mobility policy. For example, typically database components of a service are unlikely to move. Other factors are external related to service management and logistics. For example, load condition of the servers at the serving network, the service hosting policies of the serving network, and the contract between the serving network and the service provider are some external factors that play a role in the decision of the final mobility of the service components.

The network support is required to move services across provider domains. Before moving a service some sort of agreements must be in place between the network operator and the service provider. A handshake protocol is described below to move a service dynamically. The network operators should have some kind of service platform that is capable of hosting mobile services.

2.4 Service Mobility Protocol

The request to move a service comes from a network operator to the service

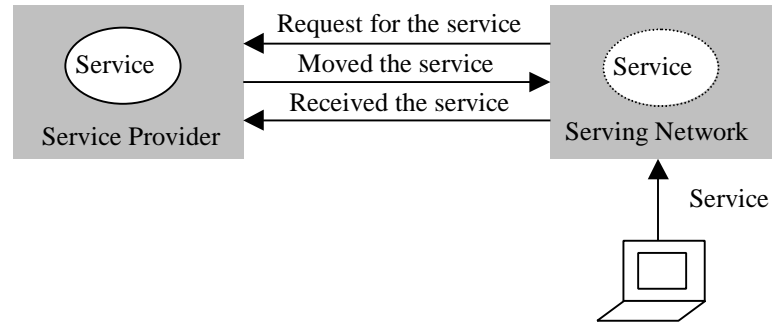


Figure 1: Service Mobility Protocol

A *service level agreement (SLA)* between the network operator and the service provider must be in place prior to establishing a connection for the service. The service level agreement contains some agreed upon service parameters between the network and the provider. It may also contain some service management information, such as whether the network operator can offer new subscriptions for that service, time to lease the service if it is on lease, etc. The network operator sets up one of its servers to receive the service components before sending a *request for a service* message to the service provider. After receiving the request, the service provider creates a mobile instance of the service based on the SLA, and moves the service components to the server assigned by the network operator for receiving the service. Once all the mobile components of the service are moved to the new network and assembled to form the service, the service provider receives a signal from

service to the ready to serve state. At this point the network operator sends a *received the service* message to the service provider to conclude the service mobility handshake.

3. A Mobile-Service Architecture

A mobile service can be accessed from different types of terminals in different network environments. The service presentation needs to be adjusted for different terminal types without affecting the basic service functionality. Hence, the presentation logic must be kept separate from the service core logic. The service presentation logic or *service user interface* (service UI) resides at the terminal, which is used to access the service. For full-scale service mobility, both the service core logic (or the *service*) and the service UI must be mobile. In the architecture presented in this section both sub-systems are mobile, but they are discussed separately.

terminals only the front end resides there. The service UI can be downloaded from the service provider on the user's request. It can also be a value-added service obtained from a third party value-added service provider. For example, the directory service provider can provide a directory service, but to access the directory from a specific terminal, say an SMS terminal, an independent SMS gateway service provider may provide SMS gateway service.

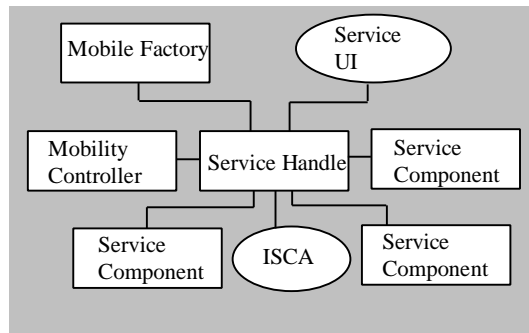


Figure 2: A mobile service architecture

Services vary in their computation, storage and networking requirements. There are two major types of services - *multi-user* services and *single-user* services. Typically, multi-user services are multi-threaded and open many simultaneous connections with multiple user terminals. These services also keep a substantial amount of state information. The single-user services

with asymmetric connections, meaning the bandwidth of the connection between the user terminal and the server hosting the service is different in both directions. For example, typically large amount of data transfer takes place from the server to the terminal, hence requires more bandwidth than the channel from the terminal to the server.

The components of a service that can be moved to a serving network vary from one service to another. For some services, all the components are cloned and bundled into a service package, and then deployed to the serving network by the service provider. This mobility policy may be suitable for some *lightweight* services. The service provider for such a service can keep an inventory of service instances that are ready for deployment in serving networks. If a service is designed to be reusable and a serving network acquires the service on lease, it may return the service to the service provider at the expiration of the contract, which can be deployed to another network on demand. For some *heavyweight* services not all components are mobile. Some components of a service instance created by the service provider may stay in its network, whereas others may move to the serving network. For example, database component of a service typically stays in the provider network. This kind of arrangement requires live connection between the two networks

section is its flexibility in allowing services to have their own mobility policies and in meeting a broad set of service requirements ranging from light-weight to heavy-weight services. Figure 2 shows the block diagram of this architecture. The architecture is an extension to the simple service architecture discussed in section 2.2. It enhances the functionality of the service factory and service handle, and adds new components to the architecture. In addition to creating non-mobile service instances, the service factory is also responsible for creating mobile instances. When it is contacted for a service instance, depending upon the situation, it may return an already created instance from the inventory or create a new service instance. The service handle API is extended for moving a mobile service instance. The service mobility controller is introduced in the architecture to handle all the activities related to service mobility. When a service provider receives a request for the service from the serving network, it gets a mobile service instance from the service factory. It then calls the service handle to move the service when the service is in the *ready to move* state. This call activates the service mobility controller to move the service, and also changes the service state to the *moving* state. The mobility controller moves the components that are already created and assembles them at the serving network. During the *assembling* phase inside the

moved. The serving network registers the service in its domain, and interacts with the service mobility controller (through the service handle) to setup the service according to the new environment. During this phase the serving network may call the mobility controller to distribute the service components across multiple servers. The service distribution strategy is separate from its mobility strategy. The mobility controller only controls the mobility strategy, but the serving network dictates the distribution strategy. The mobility controller, after completing the service setup, changes the service state to the *ready to serve* state.

The mobile factory is an optional component, because some services may not need it if its components are not created in the serving network. Some services may have database as one of their components. Databases are typically immobile. For such services, a database proxy is created and deployed at the serving network to handle the database access. The proxy makes remote calls to the database component. If a service needs to be connected to a local database at the serving network, a proxy for that database is created to handle the local database accesses. The services exchange information amongst each other through Inter-Service Communication Agent (ISCA).

A service goes through several states in

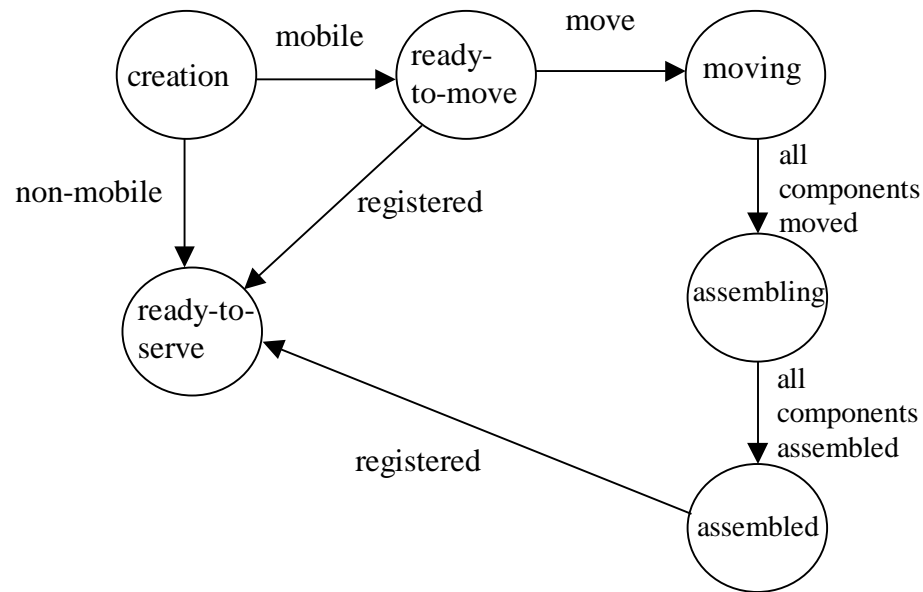


Figure 3: Service state diagram

it is moved to the serving network. While the components of a mobile instance are being moved to the serving network, the service remains in the *moving* state.

Once all the components are moved, they are assembled at the serving network in the *assembling* state. After the components are assembled the service remains in the *assembled* state unless the service is registered with the serving network and becomes *ready to serve*. Figure 3 shows the service state

Virtual Address Book (VAB) service is an example of a personal service. It manages the address books of users, and provides access to them from any terminal and through any network. A VAB service can be designed to store the data inside the network, but a user may not want to store some of his address books at a server in the provider domain for privacy and security reasons. He would prefer to store the addresses at his own terminals.

In this section, we describe the design

implemented to achieve information consistency across address books that are stored in many terminals; some of them may be disconnected from the network. Finally, we will discuss our implementation of Inter-Services Communication Agent (ISCA) to exchange information between VAB and other services.

4.1 VAB Design

The main features of our VAB service design are the distribution of computation between user-terminals and provider-servers, service mobility, information consistency in a disconnected environment, and Inter-Server Communication Agent (ISCA). Figure 4 shows the design of a VAB based on the mobile-service architecture discussed in Section 3. In this design, address books can be stored in user-terminals as well as in provider-servers. Address books can also be replicated on multiple terminals. All replicas are updateable. The VAB systems running on user-terminals are called *terminal VABs*, whereas those running on provider-servers are called *provider VABs*.

The *storage manager* is responsible for storing and managing local access to address books and their replicas. It stores the data in XML files, which is discussed below. The VAB systems must include storage and transaction

systems also include *transaction manager*. The transaction managers coordinate all the address book access activities from receiving the user request to returning the result. The provider VABs also include *transaction managers* handling remote transactions. For example, when a user wants to retrieve entries from an address book and the entries are not available locally, then the remote transaction manager brings the entries from the provider network and stores them in the *cache manager*. The cache manager can be located at either the user terminal or the provider server. In our current implementation the transaction manager does not provide all the ACID properties. Replication control and synchronization are performed by the VAB systems running on provider servers. Thus, replicas can be synchronized even when the terminals modifying address books are disconnected from the network.

Replica control and synchronization are performed with the help of the *terminal agents*, the *replica agents*, and the *VAB locator*. The location information of address books and their replicas are stored in a LDAP directory located in the service provider domain. The VAB locator in a provider VAB accesses the directory for information pertaining to the VAB location. There is a terminal agent in the service provider VAB corresponding to each user terminal

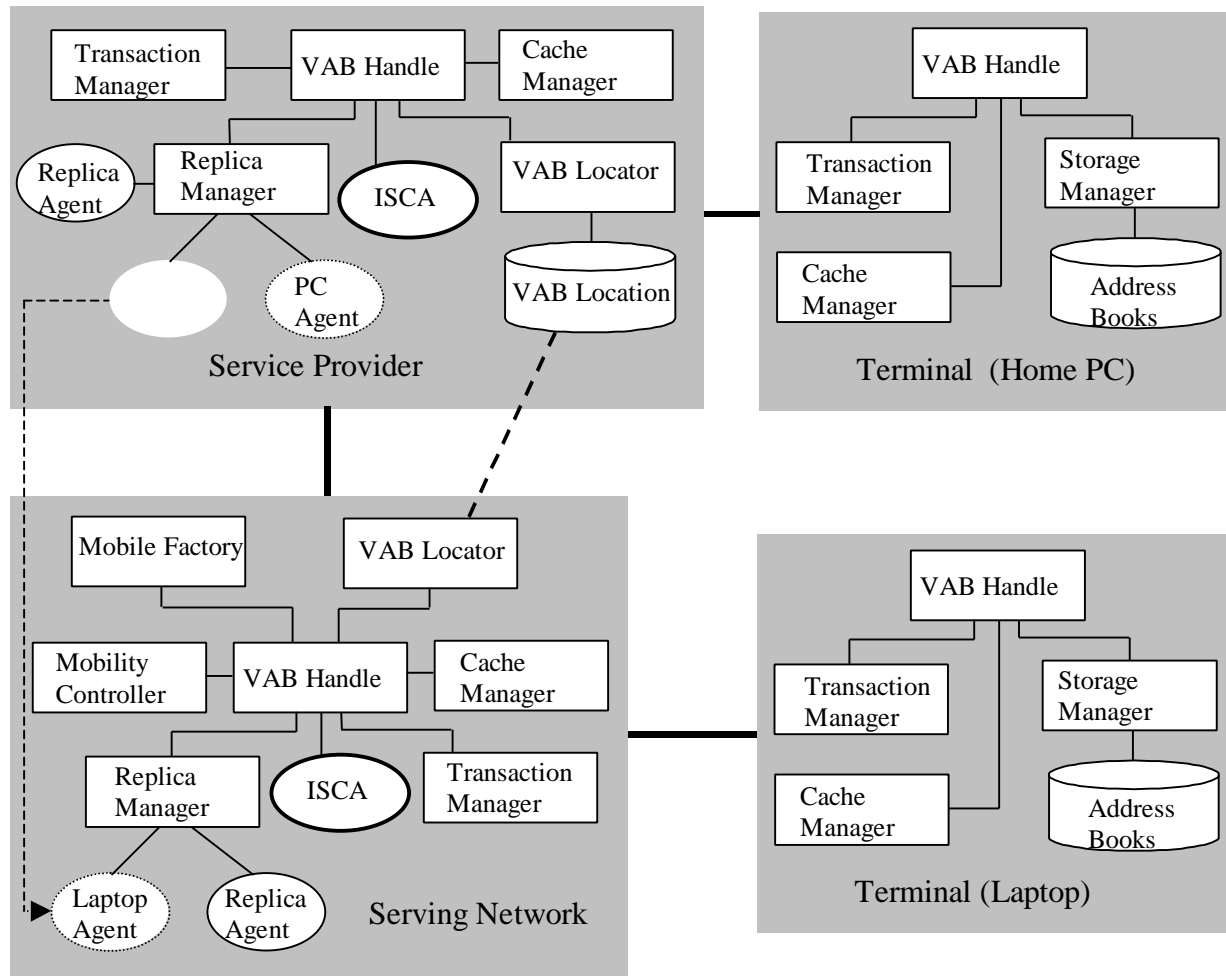


Figure 4: A virtual address book service configuration

modified, and synchronizes all the replicas.

4.2 VAB Data Storage

All data related to VAB entries may be

XML data is unmarshalled into objects (using the schema) when the data needs to be retrieved.

One reason XML was chosen as the file format is because we hope to achieve

information. It is possible that users could use web browsers (rather than the VAB GUI Application) to view their VAB data directly from XML. Finally, XML is useful for generic messaging between components on a network. Rather than using a pre-defined API to communicate, components can simply receive XML messages with the associated schema and commands. This allows VAB components to be more flexible and therefore communicate with more types of components dynamically.

There are a few disadvantages of using XML for data storage. Storing binary data in XML can be difficult. Normally, this data is stored in an external file and then referenced by the XML. This makes data mobility more difficult. Also, object to XML mapping (and vice-versa) is not very efficient. Efficiency is sacrificed for the useful abstraction provided by XML.

4.3 Information Consistency

Information consistency is harder to achieve for a personal service, such as VAB, because the data is scattered across several computers and some of them are disconnected to the network. It becomes more challenging when updates are allowed on VABs stored in the disconnected terminals. Let us consider one of the many possible scenarios to illustrate the problem unique to this kind of service. A user has two address books, a *home address book* and a *work*

provider network most of the time. Hence, all the address books stored in the home PC are accessible from any serving network. Since a laptop is a mobile terminal, the work address book is readily available to him. He keeps his PDA wherever he goes. The PDA does not store an address book, but keeps a cache of small number of most frequently used phone numbers. In his business trip he takes his laptop and PDA. While roaming in a foreign network, he wants to access his home address book using his laptop. The provider VAB at the foreign (serving) network brings him the entries from his home address book. He modifies the local copies of those entries and disconnects. Later he accesses the home address book from his PDA and wants to access those modified entries. He cannot do that unless the home address book is synchronized with the modifications stored in the laptop. He cannot access the data stored in his laptop from his PDA, because the laptop is disconnected from the network. Two problems emerge from the above scenario. One, the work address book is not accessible from all the terminals, because it is stored in the terminal (laptop) that is not always connected to the network. Two, the modifications to an otherwise available (home) address book is not accessible, because they are stored in the terminal that is not connected to the network. The first problem does not arise if the address books are stored at the provider

Let us consider the above scenario including the replica of work address book at the home PC to illustrate the solution for synchronization of address books we propose in this paper. When the user moves with his laptop and connects to the serving network, the network brings a mobile provider VAB from the VAB service provider. It also notifies the service provider that the laptop is connected to the serving network. It brings the laptop agent from the service provider VAB if the agent has data available for synchronization. The laptop agent contains all outstanding updates for the work address book that were accumulated over the period of time since the laptop was last synchronized. All the updates are then committed to the work address book after that the laptop agent is returned to the service provider VAB. Later on, if the user modifies the work address book on the laptop, copies of the modified entries are also stored in the replica agent of the serving network provider VAB to synchronize the replica of the work address book located at the home PC. Asynchronous communication is established to transmit the entries from the laptop to the provider VAB located at the serving network. This requires queuing the entries at the laptop. Thus, the terminal VAB is not restricted by any disruption of communication with the serving network [6]. The queue retains entries while the laptop remains disconnected from the network. After it

The replica agent at the serving network keeps modified entries for the home address book as well as the work address book replicas. It periodically moves the entries to the PC agent. The PC agent commits the updates to the respective address books stored at the home PC.

The synchronization solution presented above does not constrain the exact location of a terminal agent. In our implementation the exact location is kept in the directory and can be obtained through the VAB locator. The decision to move the laptop agent to the serving network (or as a matter of fact to the laptop) when the laptop is connected to it is to improve performance by committing all the outstanding updates locally.

4.4 Inter-Service Communication Agent (ISCA)

The VAB provides a distributed Address Book facility and stores information such as phone numbers and work and home address information. All CAMELEON services communicate with the VAB using an Inter Service Communication Agent (ISCA). The ISCA is a service specific agent that facilitates the VAB to communicate using FIPA [3] ACL [4]. Here the assumption is that every CAMELEON service designed to inter-operate with other services using ACL has an ISCA like component that provides

between agents. Each ACL Message contains generally used information such as the sender and intended receiver of a message. Also included is a content field the contents of, which are specific to a particular application. In the case of the VAB ISCA this is an XML [2] document defined in RDF [9], building on the schema for vcard [10]. This RDF schema is comprised of two types of information: A set of data representing the attributes of a VAB entry (e.g. phone numbers) and information relating to the purpose of the message (e.g. the action the ISCA is to initiate on the VAB upon receiving the message). The XML content allows access to a wide range of the possible actions the VAB can perform. These include inserting the VAB entry information that the message contains into a particular VAB or performing searches. Hence through the deployment of FIPA Agent based ISCA's service interoperability is achieved.

5. Concluding Remarks

In this paper we have described a service mobility architecture that allows services to be mobile across multiple provider domains. For services to be available at a foreign network, some sort of service level agreement is required to be in place in advance between the two service providers. We have also described the design and implementation of *Virtual Address Book* (VAB) service. An important feature of the VAB service

Services Communication Agent (ISCA) using ACL.

References

1. Cameleon Deliverable D03: "Specification of Prototypes". CEC Deliverable Number AC341/CN/WP2/DS/I/9812/b1. December 1998.P 66. Chapter 7: "Virtual Address Book"
2. Extensible Markup Language (XML) W3C recommendation 10/2/98, <http://www.w3.org/TR/REC-xml>
3. Foundation for Intelligent Physical Agents, <http://www.fipa.org>
4. Foundation for Intelligent Physical Agents (FIPA) "Agent Communication Languagev FIPA 97 Specification: Part 2, Version 2 [URL] <http://www.fipa.org/spec/fipa97.html>
5. Java Home Page, <http://www.javasoft.com>
6. D. Joseph, J. A. Tauber, and M. F. Kaashoek, *Mobile Computing with the Rover Toolkit*, IEEE Transactions on Computers, February 1997
7. Object Management Group, <http://www.omg.org>
8. E. Pitoura, and G. Samaras, Data

10. Renato Iannella. Representing vCard v3.0 in RDF. 22 January 1999.
11. The Agent Society,
<http://www.agent.org>
12. UMTS Service aspects, Virtual Home Environment (VHE), UMTS TR 22.70 V2.0.0 (1998-03), ETSI.