Supporting Universal Personal Computing on Internet with Java and CORBA

Jinsong Zhu[1,2], Maria Törö[1], Victor Leung[1], Son Vuong[2]
[1]Dept. of Electrical and Computer Engineering, [2]Dept. of Computer Science
University of British Columbia
Vancouver, B.C., Canada V6T 1Z4

Abstract

Universal Personal Computing (UPC) supports nomadic computing on the Internet by allowing mobile users to access their personalized computing resources anywhere on the Internet using any available terminal. Since UPC must deal with heterogeneous systems in a mobile environment, independence of computing platforms and support of portable objects are imperative, which leads to the choice of Java. The system is modeled as distributed objects with CORBA as the common bus that facilitates distribution, interfacing, and integration of the objects. The combination of Java and CORBA constitutes an ideal environment for UPC.

1. Introduction

Nomadic computing on the Internet has emerged as a new research area due to the popularity of the Internet, the widespread use of portable computing devices such as personal digital assistants (PDAs) and laptops, and users' desire to be networked. In such an environment, nomadic computing should provide independence of location, of motion, of platform and with widespread accessibility to remote files, systems and services [2]. In other words, the essence of such a computing paradigm is to retain one's personal computing environments and access capabilities wherever one happens to be. Typically, this is achieved by physically carrying a computing device such as a laptop or a PDA with appropriate Internet connectivity (via wireless or wired links) and mobility support such as mobile IP [7].

We envisage another form of nomadic computing on the Internet in which it is the user's personal computing environment that is mobile. A personal computing environment consists of the set of resources, services, and personal preferences as configured by the user at her home network. When moving, the user does not carry any computing device but instead uses any available machine where she visits and yet her familiar personalized computing environment is still available to her. Imagine, for example, in a hotel, at an airport, or even in a café, there deployed many public terminals (similar to public phones today) that were connected to the Internet for customers to use. In such a case, it is desirable that one can simply use an idle terminal to access the network as well as her own computing resources within a familiar environment, as if she was using her home machine. She should not need to worry about what hardware or software platforms she is using, just like one does not need to wonder about the brand or vendor of a public phone. We call this form of mobility *user mobility*, as opposed to *terminal mobility* where the physical device migrates [7,8]. The public terminal being used is called a *foreign terminal*, while the network that the foreign terminal is attached to is a *foreign network*. The system that supports user mobility is called a *Universal Personal Computing* (UPC) system [3,4]. UPC accomplishes a global virtual network service independent of location, motion, and platform. It is a new mobility-support service one step further from the basic terminal mobility support.

To provide a personal computing environment to the user on any available networked machine, it requires a virtually homogeneous distributed processing environment that is independent of the computing platforms. Java, whose "Write Once, Run Anywhere™" model provides a portable object infrastructure that works on every major operating system, satisfies our need. Moreover, an object-oriented model is desired as the personal computing environment consists of natural objects such as service profiles and configurations. The objects are also distributed and moving around based on the current location of the mobile user. This calls for a distributed object oriented framework. The Common Object Request Broker Architecture (CORBA) developed by the Object Management Group (OMG) provides such a framework [1,5,9]. We favor CORBA over Microsoft's Distributed Component Object Model (DCOM) because CORBA is an open standard supported by a majority of the industry and is available on most platforms. While Java Remote Method Invocation (RMI) can be a competitive alternative, CORBA provides a more generic, language-neutral object model. We therefore choose Java and CORBA as the implementation platform for our UPC system.

2. Universal Personal Computing

UPC allows global access to the Internet regardless of the user's point of attachment to the network. This global access includes the ability to use services that the user wishes to bring with her, and to access her personal computing environment (PCE) where and when desired.

A portable object implementation (with Java) and a distributed object architecture (such as CORBA) are essential for UPC to work because objects in a UPC system are distributed and need to be easily accessible and movable on the network. A foreign terminal could be running any operating system, hence the independence of operating systems is crucial. Java nicely supports such a dynamic computing paradigm. With the support of the Java Virtual Machine or a Java-enabled browser, both standalone Java applications and applet based applications can be moved and run at any foreign location supporting UPC.

A simplified UPC architecture is presented in Figure 1. The foreign terminal or its associated host is assumed to be able to host a terminal agent and a Java virtual machine, and to support a graphical user interface.

From an object-oriented view, the UPC system consists of the following major objects: user agent, user profile, terminal agent, terminal profile, and facilities. There is a process, called initial agent on the foreign terminal, that provides an interface to the mobile user, and a standalone application in the home network that manages the PCE, allowing mobile users to register, update or delete their PCEs. The PCE manager itself may also be brought to a foreign site allowing a mobile user to update (but not to create or delete) her PCE on site.

The terminal agent resides in any foreign terminal that supports UPC, while the user agent sits at the user's home network and keeps track of the user profiles. When a user logs into a foreign terminal with her logical user identifier (LUI) and password, the initial agent calls the terminal agent to locate the user agent for the user, authenticate the user, and obtain the set of home services that the user wishes to bring in (which has been registered in the PCE in advance). This set is compared with the terminal profile to determine the services that this terminal is capable of supporting, in terms of, say, terminal hardware constraints such as colors and resolutions. The terminal agent also checks with the local service providers to see if the above services are also available locally, and gives the user a choice of using it locally if available.

The user interface will display two set of services: 1) those available at home, and capable of being supported by the foreign terminal; 2) those available at the foreign network locally. The user can then choose the service she wants, and the system will bring in the PCE as required and run applications (download first if necessary) to provide the desired service. The user interface also allows the user to query the local service provider for any other locally available services or facilities, based on the service types and attributes that she specifies. This accomplishes our virtual global UPC network.

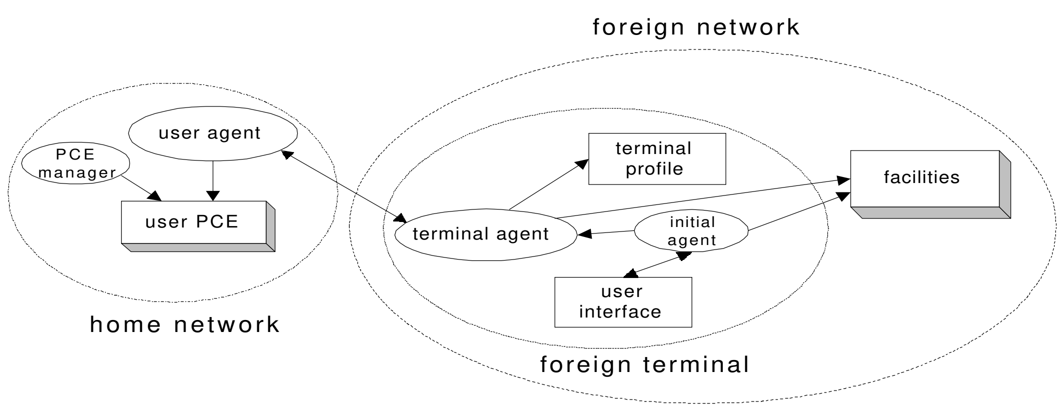

Figure 1. Major components of the UPC system

3. Supporting UPC with Java and CORBA

3.1 Using CORBA in UPC

The Common Object Request Broker Architecture (CORBA) defines a middleware that bridges distributed objects across heterogeneous environments. It allows client objects to invoke server objects across the network without dealing with the underlying complexities of object implementation and invocation. All objects, as long as they expose a well-defined interface in the *Interface Definition Language* (IDL) that describes the services they provide to other objects, can be invoked anywhere in the network. The implementation of the object (language, operating system, or any other system aspects) is not visible nor of interest to the caller. The calling object does not even have to know the location of the invoked object provided the object reference is known. This model fits well into UPC since it essentially provides a homogeneous distributed processing environment.

CORBA also defines a wide range of services (CORBAservices [6]) and facilities (CORBAfacilities) to extend the core capabilities of the ORB. These services are important for the UPC system since they make many aspects of UPC easier to implement. Currently we only consider using the CORBAservices summarized in Table 1 in our UPC system.

Since not all CORBA services are available in existing ORB products, in our current prototype we are only able to use the Naming and Trader services. The Naming service has proved to be the most essential one since we need to use it to locate objects. Unavailable services are temporarily substituted with our own code.

Table 1. CORBA services utilized in UPC system

Object in UPC	CORBA services used	Functionality
PCE	Naming	Registration of PCE
	Persistent object	Storage of PCE state
	Relationship	Association of PCE components
	Life cycle	Copying/moving/deleting of PCE component objects
User Agent	Naming	Association of User Agent and LUI
	Relationship	Extracting appropriate part of PCE
	Concurrency control	Simultaneous access to PCE
Terminal Agent	Security	User authentication at login and for use of local facilities
	Naming	Lookup for User Agent by LUI
	Life cycle	Copying PCE components
	Trader	Lookup for facilities at foreign network

3.2 Java in CORBA

Java extends CORBA nicely by allowing dynamic deployment of objects across the CORBA infrastructure. The objects can be moved from node to node. The support of replaceable Java ORBs makes it possible for Java code to be independent of a particular ORB. Java is also an object-oriented language that fits naturally into the CORBA object orientation architecture.

For the UPC system, Java on CORBA is especially important since a foreign terminal may be running any operating system. In order for the mobile user to run at a foreign site an application that is hosted at her home environment, the code must be portable and independent of the operating system. Java's "Write Once, Run Anywhere" feature satisfies this requirement. This portability of Java objects makes it possible for UPC to furnish not only a transparent static computing environment but also transparent computing services provided by applications. The combination of Java and CORBA also allows Java applets to interface with CORBA within a Web browser.

Note that we choose to use Java to implement UPC objects and require any applications that may be run transparently at a foreign terminal be written in Java. However, the server objects on the application server do not have to be written in Java. In fact, CORBA allows legacy code to be wrapped in objects so that they can be made usable by other objects or programs.

4. UPC System Object Classes

In this section, we define the major object classes in the UPC system, in order to better describe the functionality of the system. Referring to Figure 1, these objects are: (a) those residing in the user agent machine of the home network: user's PCE and user agent; and (b) those residing in the foreign terminal: terminal agent, terminal profile, and the user interface provided by an initial agent.

In CORBA, server object interfaces are specified in the Interface Definition Language (IDL) which is part of the CORBA specification. We use Java/IDL mapping in our system.

4.1 Personal Computing Environment

A PCE is defined for each user. It consists of a set of objects that the user wants to use when being away on a foreign network using a foreign terminal. Examples of such objects include: application objects that provide certain services, resource objects such as files, and environment objects that define service configurations and preferences. The PCE objects will be managed by the User Agent (UA). Mobile users are required to provide and register their PCEs with the UA.

4.2 User Agent

The UA is an object that manages the PCE objects for users in a domain and also communicates with the Terminal Agent (TA) in authenticating a mobile user and transferring the PCE to the foreign terminal. Each UA is registered in the CORBA Naming Service with a binding to a domain name. This allows the terminal agent to locate it after a user logs into the foreign terminal using an LUI containing the domain name.

The operations of a UA provide methods for the TA to authenticate a user, to update a user's location (machine name or IP address), to get and update the PCE of a user, and to get the profile by application name. A special `logout()` operation is called by the TA when the user finishes a session and logs out from the foreign terminal. This allows the UA to unbind the user from its current location.

4.3 Terminal Agent

The TA object is responsible to handle user requests at the foreign terminal, called by the initial agent in response to user requests. The first operation called is the `login()` method for authenticating a mobile user. After login, the TA is responsible for determining an acceptable subset of the user's PCE as supported by the terminal capabilities specified in the terminal profile. It also discovers the locally available facilities by inquiring the local Trader Service when requested by the user.

The `login()` operation takes a user's LUI and password as parameters. From the domain name part of the LUI, the TA can determine the location of the UA by using a directory service. The object reference for the UA can then be obtained by using the Naming Service as mentioned before.

The operation `run_app()` runs an application. If the application is locally available, the TA simply runs it locally, with appropriate configurations set for it; otherwise this operation fetches the application and its environment from the UA and then runs it. The operation should be an asynchronous one, in that the caller (initial agent) can resume operation after the call is issued without waiting for its completion. This allows the initial agent to accept user's input while the application is running.

4.4 Terminal profile

The terminal profile defines the capability of the terminal. This includes the physical capabilities of the screen such as colors, resolution and size, and software capability such as available applications on the terminal (i.e., directly accessible from the terminal). It also includes mapping rules for application configurations.

The mapping is typically a matter of copying the configuration files (part of PCE) to the local file system where the application will find them. The implementation of this mapping will be dependent on the operating system of the foreign terminal and must be defined and implemented differently for each platform that the TA runs on.

5. Conclusions

We have presented a universal personal computing system on the Internet that supports nomadic computing. The core of the UPC system is the support of user mobility, i.e., regardless of where the user is connected to the Internet and what machine is being used, the user can enjoy a familiar computing environment that she has specified. The use of Java and CORBA allows us to model and implement the system in an object-oriented manner, and more importantly, to solve the portability and heterogeneity problems which are inherent in a distributed mobile environment. The CORBA services in particular help in dealing with object distribution, interfacing and integration.

We consider UPC as an effort in realizing ubiquitous computing [10] on the Internet by making user's computing environment available on any Internet-connected machines. The only requirement is that the machines support Java and CORBA, which should be reasonable considering the widespread use of distributed object-oriented computing on the Internet.

Our future research will further evaluate the UPC system, its protocol architecture, performance, and security management. Further directions in ubiquitous personal computing on the Internet will also be explored. We have implemented a prototype using Visigenic's VisiBroker for Java on Solaris 2.x. This prototype will be used as a testbed for our future work.

6. Acknowledgment

This work was supported by a grant from Motorola Wireless Data Systems Division, Richmond, B.C., Canada. The authors wish to thank the anonymous reviewers for their helpful comments.

7. References

1. Baker, S., Cahill, V. and Nixon, P. (1997). Bridging boundaries: CORBA in perspective, *IEEE Internet Computing*, September-October, 1997.
2. Kleinrock, L. (1995). Nomadic Computing - An Opportunity, *ACM SIGCOMM Computer Communication Review*, Vol. 25, January 1995.
3. Li, Y. and Leung, V. (1997). Supporting personal mobility for nomadic computing over Internet, *ACM Mobile Computing and Communications Review*, Vol. 1, No. 1, April 1997.
4. Li, Y. and Leung, V. (1997). Protocol architecture for universal personal computing, *IEEE Journal of Selected Areas in Communication*, October 1997.
5. Object Management Group. (1997) The Common Object Request Broker: Architecture and Specification, V2.1, September 1997. Available at http://www.omg.org.
6. Object Management Group. (1997) CORBA Services Specification, December 1997. Available at http://www.omg.org.
7. RFC 2002. (1996) IP Mobility Support, C. Perkins, Editor, October 1996.
8. Teraoka, F., Yokote, Y. and Tokoro, M. (1991). A network architecture providing host migration transparency, In *Proc. SIGCOMM'91*, September 1991.
9. Vinoski, S., (1997). CORBA: Integrating diverse applications within distributed heterogeneous environments, *IEEE Communications Magazine*, February 1997.
10. Weiser, M. (1993). Some computer science issues in ubiquitous computing, *Comm. ACM*, July 1993.