CISIS 2011
The Fifth International Conference on
Complex, Intelligent and Software Intensive Systems

Korean Bible University, Seoul, Korea
30 June - 2 July 2011

Edited by
Leonard Barolli, Fukuoka Institute of Technology, Japan
Fatos Xhafa, Technical University of Catalonia, Spain
Ilsun You, Korean Bible University, Korea
Nik Bessis, University of Derby, UK
S14: Complex Systems and Software Modeling and Analytics

- A New Embedded Coprocessor for Clifford Algebra Based Software Intensive Systems
  Silvia Franchini, Antonio Gentile, Filippo Sorbello, Giorgio Vassallo, and Salvatore Vitabile

- Development of a Bluetooth Intelligent Multi-user Environment Based on Wireless CORBA
  S.L. Toral, F. Cortés, F. Barrero, and D. Gregor

- A Periodic Routing Problem with Stochastic Demands
  Michael Schedl and Christine Strauss

- Performance Evaluation of Multihopped Cognitive LANs
  Goshi Sato, Ryu Sygimoto, Noriki Uchida, and Yoshitaka Shibata
Development of a Bluetooth Intelligent Multi-user Environment based on Wireless CORBA

S. L. Toral, F. Cortés
Electronic Engineering Department
University of Seville
Seville, Spain
toral@esi.us.es, fcortes1@us.es

F. Barrero, D. Gregor
Electronic Engineering Department
University of Seville
Seville, Spain
fbarrero@esi.us.es, orlandogregor@gmail.com

Abstract—The wireless CORBA specification provides a basic framework for wireless based mobile distributed environments, suggesting a generic solution which still hides the underlying network level details from applications. However, its application over Bluetooth networks exhibits some problems when dealing with multi-user environments, like power consumption constraints of mobile devices or Bluetooth roles. This paper proposes the creation of two new CORBA services and some changes in the LTP transport layer to facilitate the management of Bluetooth multi-user environments.

Keywords- Wireless CORBA; Bluetooth; intelligent infrastructure; multi-user environments

I. INTRODUCTION

The fast growth of wireless technologies makes possible not only the development of new services, but also the adaptation of existing services to wireless networks, making them available where users cannot connect to a fixed network. In this context, Bluetooth is fast emerging as the standard for short-range wireless communication. Due to its low cost, low power consumption and bandwidth, Bluetooth has become a suitable technology for building multi-user environments [1].

On the other hand, middleware technologies represent a good solutions for developing these services as they are platform-independent and take care of communication issues, thus, making development of new services easier. CORBA is one of the middleware's flagships, a mature technology widely used with many implementations on different platforms [2]. In particular, we found Wireless CORBA very suitable for our purpose since it can be deployed over Bluetooth networks. Although Wireless CORBA has some interesting mechanisms such as handoffs, it is mainly intended to be used in networks where the terminals are moving inside a certain network [3]. In fact, it provides a Home Location Agent which helps routing CORBA messages to the registered terminals in some situations.

Due to the public nature of the proposed application, there are some issues that Wireless CORBA does not take into account. For instance, in multi-user environments, it is not known a priori how many terminals are going to be connected to a network; on the contrary, terminals do not know the number of active hosts in the network or which services they are offering. Consequently, we have to consider a different scenario where “homeless” terminals (as they appear in Wireless CORBA terminology) are getting in and out of the network's area and possibly moving to other networks offering completely different services. This situation requires some slight changes in the ORB's implementation and the development of new CORBA services for helping navigation through this kind of networks. In this paper, an enhanced version of wireless CORBA is proposed to solve multi-user scenarios.

II. WIRELESS CORBA OVERVIEW

CORBA is a framework of standards and concepts for open systems defined by the OMG (Object Management Group) [4] to create distributed client–server applications. Methods of remote objects can be invoked transparently in a distributed and heterogeneous environment through an ORB (Object Request Broker). Basically, the ORB handles communications between clients and servers. More specifically, the ORB is responsible for all of the mechanisms required to find the object implementation for the request, to prepare the object implementation to receive the request, and to communicate the data making up the request [6]. Communications are based on the method invocation paradigm such as Java RMI. CORBA manages the heterogeneity of the languages, computers and networks. For interoperability purposes, the CORBA model defines a protocol for inter-ORB communications. The server interfaces are defined with a neutral language called IDL (Interface Definition Language). This language defines the types of objects according to the operations that may be performed on them and the parameters to those operations [7].

To make an application, the designer has to describe the server’s interface in IDL. The compilation of this description generates stubs (client side) and skeletons (server side), Figure 1. The stub’s role is to intercept client invocation and to transmit it to the server through the ORB. The skeleton’s role is to receive the client’s invocations and to push them to the service implementation [8]. Stubs and skeletons may be generated in different languages. For example, a stub can be in Java whereas a corresponding skeleton is in C++. The IDL language mapping is defined for many languages such as C, C++, Java, Ada, Python. This is one of the main advantages
of using IDL, its ability to work with heterogeneous environments and languages.

Client invocations are received at the server side by an adapter (the Portable Object Adapter) that delivers requests to the adequate object implementation, Figure 1. Stubs and skeletons are not mandatory. Dynamic invocations are possible on the client side (Dynamic Invocation Interface or DII) or on the server side (Dynamic Skeleton Interface or DSI). Hence, it is possible to dynamically create, discover or use interfaces.

![Figure 1. CORBA architecture.](image)

A user-written client application can invoke a method of a CORBA object by issuing a request to the object through the IDL compiler generated client stub, which is the local representation of a CORBA server, Figure 2. From the viewpoint of the client side, the object location, local or remote, is transparent. However, a client always needs to get an object reference to the server to make an invocation. Receiving requests and preparing replies in the server side is similar to the client side, but using through the IDL compiler generated server skeletons, Figure 2. The simplest implementation of a POA exports a public interface of an object implementation as a reference, and a private interface as a skeleton.

The CORBA specifications define the General Inter-ORB Protocol (GIOP) as its basic interoperability framework. GIOP is not a concrete communication protocol that can be used directly to interact ORBs. Instead, it describes how to build reply and request messages as well as other control messages and how to create and fit a particular transport protocol within the GIOP framework. GIOP assumes the underlying transport protocol is connection-oriented, full-duplex, symmetric, provides bytestream abstraction, and indicates disorderly loss of connection. The list of assumptions matches the guarantees provided by the TCP/IP protocol stack [7]. GIOP realization over TCP/IP is Internet-IOP (IIOP) and for an ORB to be CORBA compliant, IIOP must be supported, Figure 2.

![Figure 2. CORBA communication model.](image)

As a key concept for location transparency, CORBA has introduced the concept of Interoperable Object References (IORs). An IOR is a specialized object reference. It is used by an application in the same way that an ordinary local object reference is used. An IOR allows an application to make remote method calls on a CORBA object. Once an application obtains an IOR, it can access the respective CORBA object via IIOP, the protocol used between the CORBA infrastructures on both sides to exchange the request and reply messages of the method invocation on the remote object [9].

![Figure 3. Protocol stacks.](image)

Most applications make use of CORBA's capabilities over fixed networks, e.g. LAN, WAN, etc. [10], but there are some initiatives focused on adding wireless support to CORBA. As a result, the OMG released the “Wireless Access and Terminal Mobility in CORBA Specification” [11]. Some researchers, institutions and companies, along with the OMG, contributed to the development of the specification. During this process and after the release of the specification, two projects achieved an implementation of Wireless CORBA based on an open source ORB called MICO [2]. The Vivian project (http://www-nrc.nokia.com/Vivian/) developed the LTP transport layer, a L2CAP-based layer that allows the exchange of CORBA messages over a Bluetooth link providing a transport protocol that fulfills CORBA's requirements. Figure 3 shows the protocol stack using Wireless CORBA and the protocol stack using conventional TCP/IP networks.

This work was the base for another project developed at the University of Helsinki which added to MICO some CORBA services to handle these wireless connections.
providing new interesting capabilities such as handoff, or connection recovery. The resulting MICO's variant was called MIWCO although its changes were later incorporated into MICO's official source code.

Figure 4. Communication model.

Figure 4 describes the relationship between these services and regular CORBA objects. The mobile terminal connects through a GIOP tunnel to the fixed network. The endpoints of this tunnel are called Terminal Bridge (TB), on the terminal side, and Access Bridge (AB), on the network side. Circles represent ordinary CORBA objects. The bridges communicate using the GIOP Tunneling Protocol (GTP). The Home Location Agent (HLA) is responsible for tracking its terminals as they move from an Access Bridge to another. The Home Location Agent also uses location forwarding to redirect invocations intended for objects on terminals to their proper addresses. The HLA is an optional component and despite some limitations, bridges can work without it. In this case, terminals are called “homeless” terminals.

Although this communication model is valid in multi-user environments, a deeper analysis shows some problems:

- Since terminals and access bridges do not know each other a priori, a discovery process must be continuously running. Mobile terminals usually have stricter power consumption constraints; therefore, terminals should not carry out this process. Access bridges should do it instead.
- A mechanism for providing some basic information about the fixed network before establishing the first GTP connection when the terminal gets into the Wireless network's area should be available. In this basic information there may be, for example, a trading service's IOR so that the objects in the terminal can browse the services offered by the fixed network and connect to them in case the terminal decides to do it.
- There should be some kind of connection manager in both sides controlling connection issues such as Bluetooth role (master or slave), parked links, etc. and solving higher level conflicts e.g. a situation where a terminal is connected to a network and overlaps another network's area.

III. WIRELESS CORBA ADAPTATION TO MULTI-USER ENVIRONMENTS

MICO (www.mico.org) has been selected to perform the adaptation of Wireless CORBA specification to multi-user environments because it is the only known wireless CORBA implementation. In particular, version 2.3.13 has been used. The objective consists of solving the previously detailed concerns. MICO is available under the open source model and includes most of the novel features of CORBA.

Two changes have been introduced to deal with the addressed limitations. The first change consists of modifying the LTP transport layer (Figure 5), using as a starting point the one implemented by the Vivian project, called MICO::LTPTransport. Several auxiliary classes emerge from this class:

- MICO::LPTAddress. This class stores addresses from LTP protocol and it is inherited from the generic class CORBA::Address. Consequently, it can be managed by upper classes in a similar way to an ordinary TCP/IP address.
- MICO::LTPAddressParser. This class is inherited from CORBA::AddressParser, and performs the LTP addresses interpretation.
- MICO::LTPTransportServer. This class is inherited from CORBA::TransportServer and it is essentially used in the side working as a server of LTP requests.

Although the LTP layer plays an important role in wireless CORBA, two important limitations can be mentioned in Bluetooth networks:

- The LTP layer does not solve well the lack of coverage problem. Receptions failures are considered as errors that are not notified to upper classes. The results is that reading operations are continuously retried, despite of the connection being broken. In these situations, both the AccessBridge and the TerminalBridge fall into a infinite loop, increasing considerably the CPU load.
- The second limitation is also related to the lack of coverage. By default, the necessary time to detect the connection is broken is 20 seconds. This value is not acceptable in multi-user environments.

To overcome these limitations, a new ELTP (Enhanced LTP) layer is proposed. First, ELTP performs a strict control
about error messages generated by L2CAP socket operations. In particular, a special consideration is given to ETIMEDOUT error, which is the error generated when the socket is notified about a broken connection. This way, upper layers may be notified and they can finish broken connection in lack of coverage situations.

With respect to the broken connection time detection, it may be configured using a parameter belonging to the HCI layer of the Bluetooth stack. This parameter is called LST (Link Supervision Timeout) and must be configured each time a connection is created. For this purpose, a public method setlst() has been added to the class MICOLTP::ELTPTransport to reduce the broken connection time detection. Each time the class MICOLTP::ELTPTransportServer accepts a new connection, the LST parameter is configured. The modification of the LST parameter is done by the AccessBridge, which is the master of the Bluetooth network.

The second change is related to the connection managers. A topology based on connection managers both in the side of the terminal and the infrastructure is proposed. This model makes possible service networks with a set of distributed beacons providing services to mobile terminals through a Bluetooth link. From the viewpoint of the infrastructure, services may be located on any device connected to the network. Figure 6 illustrates the proposed topology. In multi-user environments, terminals are usually nomadic devices and they are not attached to a particular network. Consequently, the Home Location Agent (HLA) developed in the original wireless CORBA specification is not necessary. Instead, two connection managers have been developed in MICO: the Access Bridge Connection Manager (ABCM) in the infrastructure side and the Terminal Bridge Connection Manager (TBCM) in the terminal side. Two channels are set between infrastructure and terminals. The first one is the link between the Access Bridge and the Terminal Bridge, where CORBA request are transmitted using L2CAP layer. The second channel is specifically used for the managers’ communication, and makes also use of L2CAP layer but using a specific protocol called Network Control Channel (NCC). The interactions among processes have been designed using CORBA interfaces. Figure 7 details the interactions among processes and devices.

A. Terminal Bridge Connection Manager (TBCM)

The TBCM is responsible of setting and releasing wireless connection on the terminal side. For this purpose, connect() and release() methods of the TerminalBridge CORBA interface are used, as well as the new included methods idle_bridge() or get_strength() that provides information about the bridge activity or the quality of the wireless connection, respectively. TBCM can also subscribe to the TerminalBridge event channel for obtaining information about bridge availability or broken connections. On the other hand, TBCM also creates its own event channel for notifying applications about available connections or new access points. TBCM also includes a CORBA interface with two types of method: those invoked by NCC servers and those invoked by client applications. Figure 8 details the TBCM class diagram.

The class tmev_consumer is used for the subscription to the TerminalBridge event channel. The TBCM CORBA interface is implemented by the tbcm_impl class. The apinfo class is used for storing and exchanging network information. It has been designed to transmit and receive information using NCC protocol. The ninfo_supplier class is used for obtaining name and event services requested by other classes. Finally, the main class at the center of Figure 8 implements the network policy, and it is responsible for
taking decisions in response to different events coming from other modules. The system operation from the TBCM viewpoint is as follows: the NCC server is continuously listening on a L2CAP port waiting for incoming connections from ABCM. On detection, the NCC server sends the beacon’s information to TBCM through its CORBA interface so that the TBCM can make TerminalBridge establish a new connection if the Network Policy Object allows it. Ordinary CORBA applications on the terminal can retrieve that basic information through the TBCM interface although they do not use the same methods as the NCC server. This basic information includes context information and some basic services’ IORs. The TBCM also controls TerminalBridge and listens to its event channel in order to notify the applications when the link between the bridges is ready to send requests or when the link is broken.

C. Network Control Channel (NCC)

In multi-user environments, terminals are nomadic devices which do not a priori know some basic information to start interacting with the network. NCC protocol provides a mechanism to dynamically obtain such information:

- Context information. A description of the context in which the beacon is located.
- Basic references. It includes a list of pairs NameService/IOR. The basic services to guarantee the interoperation of terminals are the Name Service and the Trading Service defined in CORBA. However, some other specific services could be included in this list depending on the particular environment.
- Commands. They are included for an adequate network management.

This information is serialized and transmitted via Bluetooth using L2CAP layer.

Figure 10 illustrates the basic sequence that should be followed to access network services.
IV. APPLICATION AND RESULTS

An indoor orientation application is proposed as a practical application of the proposed framework. The system is intended for visually handicapped people. Figure 11 illustrates the scenario. Several beacons have been distributed with the purpose of guiding users to the selected destination.

![Figure 11. Scenario for the indoor orientation application.](image)

From a hardware point of view, the guidance system consists of principal beacons, auxiliary beacons and a mobile terminal belonging to the user of the system. The server application is installed on the principal beacon while the client application is installed on the mobile terminal. Basically, the server application provides two methods through which a graph is transferred to the client application. The graph is made up by a list of nodes and a list of paths. Each node corresponds to a principal or auxiliary beacon. The node list contains a set of parameters associated to each beacon, like an identifier, a description of the place in which it is located, its Bluetooth address and its orientation with respect to the geographical north. This last parameter is used to set an initial alignment. In the case of visual handicapped people, the alignment may be performed using acoustical signals. Once the user is aligned with the beacon, orientation instructions are transmitted taking into account the relative user position.

The first time terminals reach the area covered by a principal beacon, the list of nodes and paths is requested. Then, the client application makes available to the user all the possible destinations. Once a particular destination is selected, orientation messages are automatically provided to the user. The user can be detected by auxiliary beacons as he or she is moving close to them. The client application receives the basic information of auxiliary beacons (Bluetooth address) through the TBCM event channel. Using this information, the client application can determine the orientation information that should be provided or even situations in which the user has chosen a wrong way.

The scenario of Figure 11 is formed by one principal beacon and four auxiliary beacons. An orientation experiment has been performed to check the feasibility of the proposed platform. Table 1 details the obtained results. The first column is the time distribution of the inquiry window during the Bluetooth discovery process. For instance, 2:4 means 2 inquiry windows per 4 data transfer windows (the size of the window is 1.28 s).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4:4</td>
<td>30%</td>
<td>60%</td>
<td>10%</td>
</tr>
<tr>
<td>2:4</td>
<td>70%</td>
<td>20%</td>
<td>10%</td>
</tr>
<tr>
<td>2:2</td>
<td>90%</td>
<td>10%</td>
<td>0%</td>
</tr>
<tr>
<td>1:3</td>
<td>50%</td>
<td>40%</td>
<td>10%</td>
</tr>
</tbody>
</table>

Table 1. Success rate of the orientation application.

The next three columns detail the probability of detecting 5, 4 or 3 beacons, respectively. Notice that detecting the five beacons means that the defined way has been successfully completed. Finally, the last column is the probability of being detected by any beacon, considering the global data of the experiment. The best results are obtained using a 2:2 distribution, which means an inquiry window short enough to detect and connect with a terminal before leaving the covered area by the beacon.

V. CONCLUSION

A system for multi-user environments where different nomadic terminals are continuously getting in and out of the wireless network's area has been proposed. The use of an adapted version of wireless CORBA allows the terminal to browse the services they are interested in by accessing standard services such as the trading service. Since CORBA is a multiplatform middleware environment, the system can be supported on many different devices. An indoor orientation application has been proposed to illustrate the utility of the developed system.

ACKNOWLEDGMENT

This work has been supported by the Spanish Ministry of Education and Science (Research Project with reference DPI2009-07955) and the Consejería de Innovación, Ciencia y Empresa (Research Project with reference P07-TIC-02621).

REFERENCES


