Using CORBA to Support Terminal Mobility

Mika Liljeberg
University of Helsinki
Department of Computer Science
P.O.Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki, Finland

Mike Evans
University of Plymouth
Network Research Group
School of Electronic
Plymouth, United Kingdom

Steven Furnell
University of Plymouth
Network Research Group
School of Electronic
Plymouth, United Kingdom

Nicolas Maumon
SEMA Group Telecom
7 & 9 Rue Hélène Boucher
78280 Guyancourt, France

Kimmo Raatikainen
University of Helsinki
Department of Computer Science
P.O.Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki, Finland

Eric Veldkamp
Royal PTT Netherlands (KPN)
KPN Research
P.O. Box 421
2260 AK Leidschendam, The Netherlands

Berny Wind
Royal PTT Netherlands (KPN)
KPN Research
P.O. Box 421
2260 AK Leidschendam, The Netherlands

Sebastiano Trigila
Fondazione Ugo Bordoni
via Baldassare Castiglione 59
00142 Roma, Italy

Abstract

Nomadicity is quite a new challenge for computing and communication technologies. Some of the key issues in the field are, what is the impact of nomadicity on client-server interaction, how to support terminal mobility, and how to cope with the unique performance characteristics of wireless access. The first issue concentrates on the question of whether the computational viewpoint client-server interaction mechanisms for fixed environments remain valid in mobile environments. The other two issues are related to the engineering aspect of dealing with mobile routing and the unpredictable performance and reliability of wireless networks, generally orders of magnitude below those of wired networks. In this paper, we show a novel way to deal with these issues in a CORBA based distributed processing environment.

Object technology—CORBA, in particular—is already mature, or is at least reaching maturity. Today CORBA provides a software execution and development environment that simplifies distributed computing and application integration. The main stream of CORBA-based solutions is targeted to LAN-based applications relying on (quite) fast and reliable connections. In this paper we will show that the CORBA 2.0 specification also provides the means to support nomadic computing. We will demonstrate how the CORBA 2.0 Interoperability Architecture, together with some CORBAServices (Common Object Service Specifications), can be used to provide seamless support to terminal mobility and communication through slow wireless and wireline connections. The corner stones of our solution are mediated bridges, and an Environment Specific Inter-ORB Protocol (ESIOP) tailored for wireless networks.

1. Introduction

The ACTS project DOLMEN is developing an Open Service Architecture for an integrated fixed and Mobile environment, called OSAM [5]. One aspect that is looked at

in more detail is terminal mobility. In that area, continuous terminal mobility is seen as the most challenging issue as it places the highest requirements on the architecture.

The scope of this paper is the impact of (continuous) terminal mobility on client server interaction mechanisms over a wireless access and how this can be supported by a CORBA [4] compliant environment.

From the computational viewpoint the client server model is used to structure the interactions between objects. The mechanism used by a client to invoke an operation offered by a server comprises two steps:

- Retrieval of a reference to an instance of the interface that gives access to the desired operation.
- Invocation of the operation across the interface, using the obtained reference, provided that a valid reference has been obtained.

In that respect, terminal mobility implicitly means frequent changes of object references; in particular those of nomadic objects, because the mobile terminal and the objects contained therein are continuously on the move. This reveals the main problems of wireless access from the computational point of view:

> *Can the basic interaction mechanism, as outlined before, be used when mobile terminals are involved? Do mobile terminals impose additional constraints on the validity of interface references?*

The answers to both questions are yes and no, respectively. This paper outlines a solution in which the impact of terminal mobility on the basic client server interaction mechanism is tackled in the engineering view point by using CORBA bridging techniques.

The following requirements are imposed on the support of mobility from the computational point of view:

1. The DPE should support object interaction according to the conventional mechanism, over a wireless access network, from terminal to network and vice versa.
2. Under the assumption that a reference changes as a result of mobility, even though it points to the same interface instance, the DPE should be able to detect those changes, and to issue a new valid reference for the interface instance transparently to the client.
3. The DPE should notify interested (client) objects of the reference change.

The rest of the paper is organised as follows. In Section 2 we introduce the basic concept of CORBA, that is *bridging*, used in the DOLMEN solution for supporting terminal mobility on the DPE level. In Section 3 we describe how the bridging is used in the DOLMEN solution to solve problems due to terminal mobility. Optimisation of the CORBA Inter ORB Protocol for wireless communication paths is addressed in Section 5. Location Register and address resolution of objects are discussed in Section 6, whereas Section 7 is devoted to bridge handover and loss of signalling connection. Finally, in Sections 8 and 9 we discuss our implementation experiences and present a summary.

2. Basic Concepts

The CORBA 2.0 architecture supports "out-of-the-box" interoperability between different CORBA implementations by means of the *General Inter-ORB Protocol* (GIOP) and *Internet Inter-ORB Protocol* (IIOP). GIOP specifies a standard transfer syntax (low-level data representation) and a set of message formats for Inter-ORB communication. GIOP can be viewed as an abstract protocol, as by itself, it cannot provide complete interoperability; for actual implementation, a mapping must be defined between it and the transport used. IIOP defines such a mapping of GIOP onto the TCP/IP protocol used in the Internet. A CORBA 2.0 platform must implement both GIOP and IIOP in order to claim CORBA Interoperability conformance. However, CORBA seeks to interoperate with a wide and diverse range of different environments, and a single set of interfaces is not adequate in all circumstances. Hence, CORBA also makes provisions for Environment Specific Inter-ORB Protocols (ESIOPs). ESIOPs allow ORBs to interoperate with other objects which run in a completely different environment. For example, the DCE-CIOP (DCE Common Inter-ORB Protocol) allows CORBA objects to interoperate with objects located in an OSF DCE environment.

In the CORBA 2.0 architecture [4], *bridging* is one of the cornerstones of building interoperability support between different communication environments. A common use for an interoperability bridge is to act as a gateway between a CORBA domain and a non-CORBA environment, and translate between IIOP and a particular ESIOP designed for that environment. While ESIOPs may be optimised for particular environments, all ESIOP specifications are expected to conform to the general ORB interoperability architecture conventions to enable easy bridging.

On the architectural level, the term bridging simply means mapping between different CORBA compliant domains. In more concrete terms, a bridge resides between two domains and translates each message related to an object invocation across the domain border into a format understood by the destination domain. The translation may involve a mapping between object references, operation names, parameters, values, types, security credentials and the like. In the CORBA 2.0 specification *mediated bridging* means that the elements of an interaction relevant to a domain are first

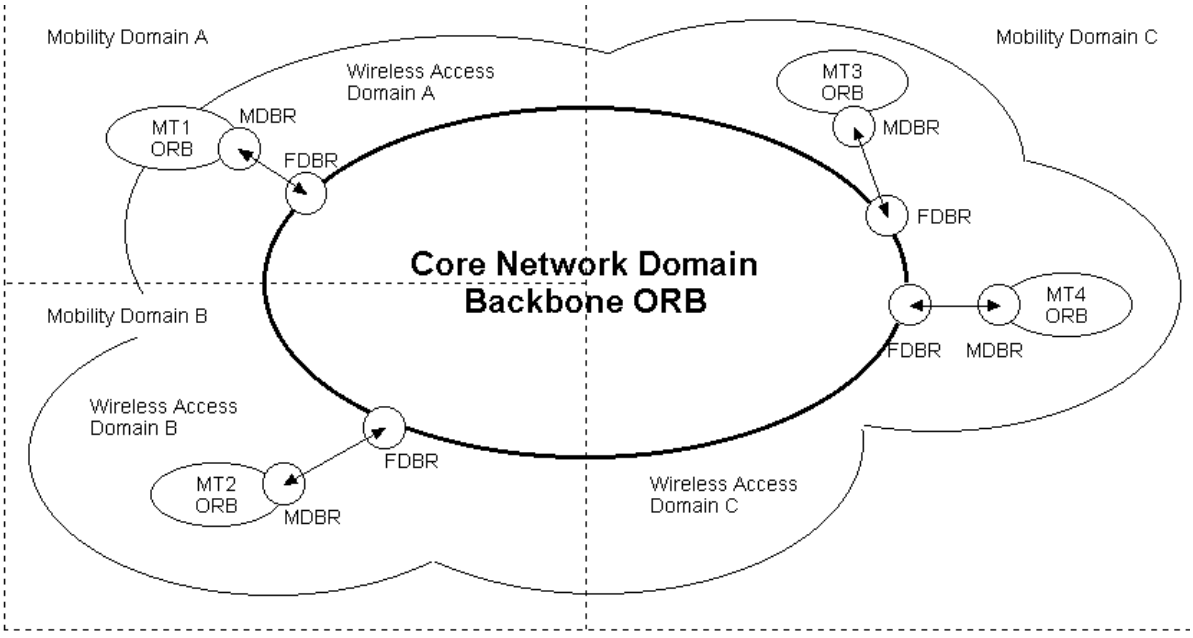

Figure 1. Bridging Mobile Terminals to a Fixed Network

transformed at the domain boundary from the internal form of the original domain to a common form of representation, then transmitted over a communication media, and finally transformed from the common form to the internal form of the new domain.

3. Building an Interoperability Bridge for a Wireless Access Network

The concept of bridging seems very appealing in its potential to interconnect mobile terminals to the fixed network. In particular, a mediated bridge would appear to be particularly well suited to the task. Implementing two half-bridges, one residing in a mobile terminal and the other in a well-known access point within each mobility domain in the fixed network, allows us to introduce an efficient light-weight Inter-ORB protocol for use over the wireless access network. This approach also allows us to address terminal mobility, performance, and reliability issues.

Figure 1 shows how mobile terminals can be connected to a fixed network domain with mediated bridging. The wireless access domain and part of the core network domain is divided into mobility domains. The core network part of each mobility domain instantiates a set of mobile-specific support services, including one or more *Fixed DPE Bridges* (FDBRs) that serve as access points to the fixed network. The rest of the core network domain serves fixed terminals and acts as a backbone network. Each mobile terminal has its own ORB that provides object services to the applications running on the terminal. Invocations of objects outside the local access domain are directed to the *Mobile DPE Bridge* (MDBR) on the mobile terminal.

The MDBR forwards the invocation to the FDBR, which then invokes the desired object. The FDBR acts as the representative of the mobile terminal within the fixed network, invoking operations in other objects on behalf of the mobile terminal. The FDBR also accepts invocation requests for objects located on the mobile terminal from objects within the core network. The FDBR forwards an invocation request to the MDBR, which then invokes the actual object and returns the response through the FDBR.

4. Light-Weight Inter-ORB Protocol

Since all object invocations between a mobile terminal and the core network pass through MDBR and FDBR, as depicted in Figure 2, the two bridges effectively form a closed interoperability domain. Thus, it is possible to transparently implement an ESIOP between the two bridges. We have taken advantage of this and designed a special *Light-Weight Inter-ORB Protocol* (LW-IOP) for wireless access networks with low-bandwidth signalling channels. The following basic requirements were identified for the LW-IOP protocol:

1. LW-IOP should be functionally compatible with GIOP.
2. Protocol message representation should be as efficient as possible.

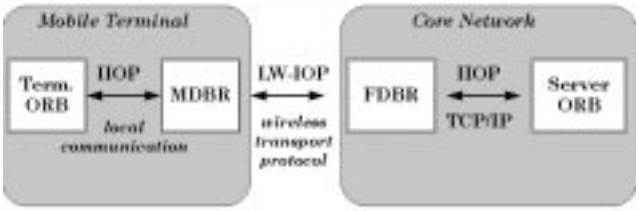

Figure 2. Protocols in Invocations over a Wireless Network

3. Redundant information in discrete protocol messages should be minimised.

The rationale for LW-IOP protocol becomes clear when one considers the design of the GIOP protocol in the context of low-bandwidth wireless access networks. GIOP was designed to be very simple to implement, and to make minimal assumptions about the underlying transport connection. The protocol itself is stateless, consisting of simple request-reply interactions. However, result of this simplicity is that each message carries a relatively large amount of redundant information that could be communicated once and then left out of subsequent messages. The data representation format is also somewhat space-inefficient.

The LW-IOP protocol is based on the GIOP protocol in the sense that the exact same functionality is supported. However, the set of messages and their representations are more efficient. In addition, caching and compression techniques have been utilised in order to reduce the number of octets sent over the wireless access network.

GIOP specifies a total of seven message types (Request, Reply, CancelRequest, LocateRequest, LocateReply, CloseConnection and MessageError). The existing structures for these messages have been transformed into a set of light-weight equivalents.

Many GIOP message elements are allocated storage which is both unnecessary and inefficient for a wireless implementation. For example, request identities are allocated 32 bits, thus allowing for the unlikely scenario of 4,294,967,296 concurrent requests. LW-IOP request ids permit 16,384 concurrent requests, and thus require only 14 bits. Other examples are message size fields, which LW-IOP reduces from 32 bits to 16 bits, and any Boolean values, which LW-IOP embeds within other data structures, rather than allocating a whole octet as GIOP does. If a client wants to send a message longer than the maximum message size on the wireless link, then LW-IOP fragments and rebuilds the message.

In message headers and bodies the basic type ‘unsigned long’ is frequently used, e.g. to indicate the length of a string. Since these values are rather small in general, a representation with a variable size is used, in 1 to 5 bytes instead of systematically 4bytes. In most cases, the numbers are represented in 1 or 2 bytes. In addition, the Common Data Representation (CDR), which is part of GIOP, inserts a lot of padding bytes for alignments of data. These are not present in LW-IOP.

In GIOP, all messages are preceded by a standard header containing a protocol identifier, version number, byte order flag and data indicating the type and size of the message. Of this, only the message type and size strictly need to be sent with each message and, as such, LW-IOP abstracts the version number into a new OpenConnection message, which is only sent at the initiation of an association, whereas other elements are discarded because they are irrelevant in LW-IOP. The byte ordering is always big endian. The remaining message type and size elements are optimised to further reduce the size of the standard message header.

Coded representations of textual and binary strings are employed to avoid the need to always transmit the full information. Two approaches are used here. Firstly, certain strings are predefined which are known to be used on a frequent basis (e.g. the name of the core network domain). Secondly, a string history is maintained through caching on each side of the link. Thus when the transmitting side wishes to resend a string that it still maintains in its cache, it simply transmits a cache reference, enabling the recipient to extract the appropriate full details from its own cache. A mechanism is used to delete the less often transmitted strings when the cache is going to be full, while maintaining cache consistency between both ends of the links. All data transmitted between caches, i.e. cache references and delete notifications, are optimised in their Light Weight Representation (LWR). A cached string can be replaced by a single byte when transmitted.

Aside from the restricted bandwidth, the other main limitation of the wireless environment is the relative unreliability of the transport medium. It is conceivable that connections will drop out or become intermittent, causing messages to be lost. As such, the LW-IOP incorporates a reliability safeguard, with client and server ends assigning a unique message sequence number as part of the message header. These are then used by an LW-IOP-specific Acknowledge message, sent by client and server, to indicate that messages have been received and may be removed from the buffer. These messages provide a mechanism for re-synchronising the transmission in the event of the signalling connection having to be recreated, as each side will know the messages that require resending.

5. Relocatable and Migrable Object References

The concepts of Relocatable Object (RO), RO identifier (ROI), and RO reference (ROR), are introduced here to

compensate for some lacks in the CORBA objects for the support of features such as mobility and migration. In brief, an RO is a CORBA interface instance which implements a TINA eCO (engineering Computational Objects) interface instance. CORBA permits a system to enhance the basic concepts towards RO concepts, while staying compliant to CORBA.

Both migrable and mobile objects are kind of relocatable objects. The main difference is that when a mobile object moves, it stays in the mobile terminal, whereas when a migrable object migrates, it changes terminal. A fixed client will perceive a mobile object as being relocatable, because when the terminal moves, the client sees it through another bridge. The common aspect between migrable and mobile objects resides in the fact that both their references become out of date after a relocation, since the route to invoke them changes. We only concentrate here on mobile objects.

It is necessary to assign an identifier to a mobile object so that it is always possible to keep track of it. The identifier, also called here ROI (Relocatable Object Identifier), must be composed of information which will never vary during the life cycle of the object. Thus an identifier does normally not contain any information on the object instance location, with the exception of elements of location information which are not variable during the life cycle. A service is needed to bind the identifier to the current location of the object instance. Such a service has been introduced in [2], under the name of Location Register (LR). The LR binds each ROI to the variable elements of location information. These elements and the ROI are the main components of a ROR (Relocatable Object Reference). Clients will use the ROR to always point to the same object instance and invoke it, even if the location changes.

In distributed applications, not all the objects instances need to be relocatable. The DPE must offer distribution transparency to the application and to the service control architecture. A client, at the application level, does not have to know if an object instance is relocatable or not in order to invoke it. A client application will invoke non relocatable and relocatable object instances in the same way. Conversely, a server, at the application level, will always know if an object it owns is going to instantiate is relocatable or not. A server application may instantiate an object and generate its reference in different ways, depending on whether it is relocatable or not. This permits a server application to attribute ROIs to the ROs it creates. When a mobile object reference enters the fixed network through the bridge, the bridge behaves like a fixed server which would have instantiated that object. The bridge translates (or create) the reference by embedding an ROI into it, since it knows that the object is relocatable. Consequently, IIOP mobile object references that are introduced in the fixed network domain by the bridge are kind of RORs.

An ROR is made by adding an ROI to the reference on the object instance which currently constitutes the RO, with the constraint that the format of the ROR remains CORBA compliant. The only possible place for the ROI in an object reference is in the object key. It is not possible to put the ROI in other GIOP fields, even if GIOP seems to permit it, because these fields are rather used for connection set up. Only the object key is transmitted in an object invocation. If the ROI was not in the object key, the server could not know if an invoked object instance is relocatable or not, and could not contribute in offering relocation transparency.

In our implementation, it is the Global Terminal Identifier (GTID) which is used as an ROI, and inserted in the object key of the IIOP object reference. Mobile objects are perceived as being relocatable only in the fixed network, and hence only the mobile terminal needs to be identified, not the exact object location into the mobile terminal.

6. Location Register and Object Address Resolution

The basic problem in mobility is that, in order to enable objects in the core network to invoke objects in the mobile terminal, the location of the mobile terminal must be known at all times. To be precise, the bridge through which the mobile terminal can currently be accessed, must always be known. The LR is a database that stores this knowledge within the core network. The address of the current primary bridge of the mobile terminal can always be retrieved from the LR if the GTID of the mobile terminal is known.

The LR is a kind of naming service which in our prototype implementation contains a persistent flat list of couples (GTID; FDBR TCP/IP address). The TCP/IP address is the one of the current FDBR through which a fixed client can invoke a mobile object instance. Thus, the LR permits a per terminal registration. If objects instances on mobile terminal would be migrable, a per object registration would be required in the LR. In this case the LR should have a greater storage capacity, and should be able to process more frequent invocations than for a per terminal registration.

The LR offers an IDL interface, with has at least the basic operations to store and remove couples of (GTID; FDBR address), and to get an FDBR address from a GTID. The LR should be distributed, and tailored for frequent updates and read accesses. The GNS and each FDBR will have to acquire the object reference of the LR interface, in a configuration step. This reference should be modified as rarely as possible.

In world-wide implementations the LR must be distributed. Most probably there will be one LR per access provider domain. This also affects the contents of LR. It should also be noted that in the home domain of the user there must be a mapping from a permanent UserId to the cur-

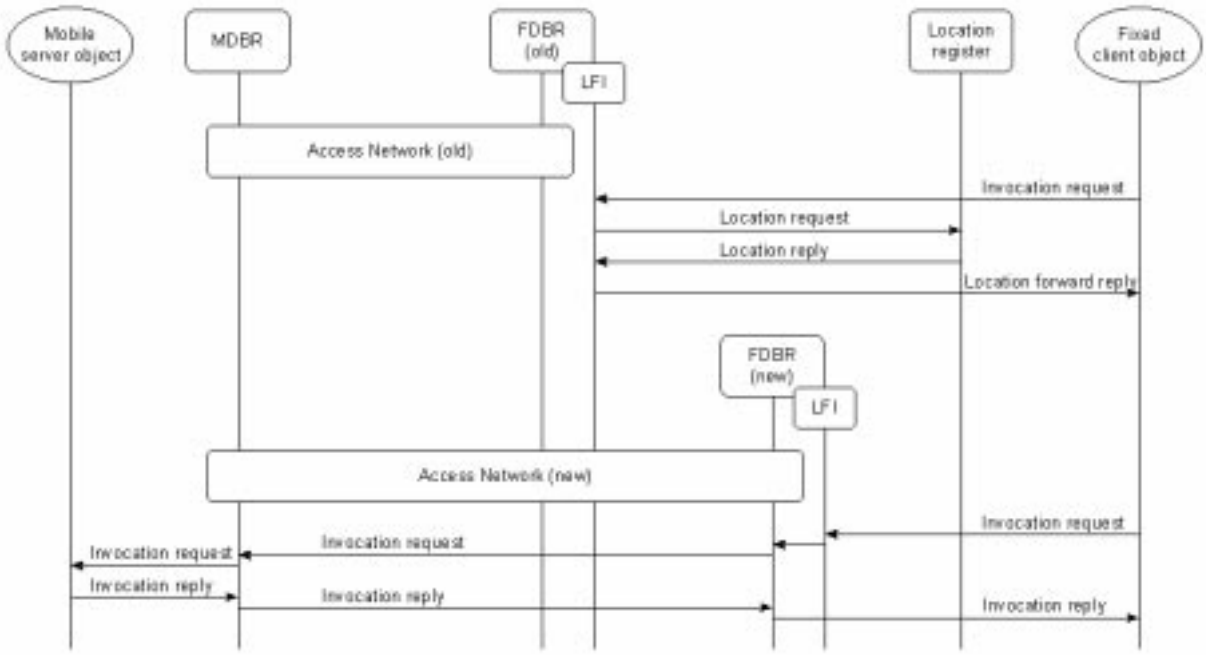

Figure 3. Message Sequence Chart

rent access provider domain so that initial access initiated from the core network would be possible.

On the DPE level there are two cases of address resolution that need to be taken into account. The simpler case is mobile-originated object invocation, where an object on the mobile terminal invokes an object within the fixed kTN network. Since the invoked object is behind the wireless access network, the invocation request is intercepted by MDBR and forwarded to FDBR. Since the FDBR is part of the core network domain, it can invoke the operation directly on the invoked object.

The mobile-terminated object invocation is the harder of the two cases. As usual, in order to invoke an operation on an object, the invoking object must first determine the object reference of the target object. However, the object reference of an object located on a mobile terminal includes a separate terminal identity (the GTID) and address. The address, when resolved, enables the complete route from the fixed client to the object to be determined. In particular, the address is composed of the local address of the object in the mobile terminal, and of the address of the FDBR, in the fixed network, which is currently bridging the mobile terminal.

As the FDBR address in the reference may become out of date after a handover, FDBRs can receive invocations for objects hosted on mobile terminals that they are no more bridging. In order to make the invocation succeed in spite of that, the FDBR includes a functionality called Location Forward Indicator (LFI), which helps the DPE to offer relocation transparency to its users. The LFI makes full use of the LOCATION_FORWARD mechanism offered by CORBA in GIOP. The principle is explained below.

When a fixed client invokes a mobile object through an old FDBR which is no more bridging the mobile terminal, the LFI in the old FDBR replies to the client proxy with a LOCATION_FORWARD status and with the up-to-date reference of the target object. This reference contains the address of the new FDBR which is currently bridging the mobile terminal. At reception of a Request, the LFI has to (see also Figure 3):

1. Extract the GTID from the object key, which is transported in the header of the request. The object key is an element of an object reference.

2. Verify if the FDBR is currently bridging the mobile terminal which hosts the target object, using the GTID.

3. If yes, the LFI allows the FDBR to transmit the request towards the mobile terminal.

4. If not, the LFI interrogates the LR (Location Register), by providing it the GTID to get the address of the new FDBR, in the fixed network, which is currently bridging the mobile terminal.

5. The LFI rebuilds a new object reference, which includes the address of the new FDBR.

6. The LFI replies to the client with a LOCATION_FORWARD status and the new reference.

7. The client replaces its old reference by the new one, and reissues the request through the new FDBR. Since all this is performed by the client proxy, it is transparent for the client application.

The only constraint of this method is that FDBRs should always remain alive to be able to answer to clients which have old references. This constraint is acceptable since FDBRs are permanently active.

7. Bridge Handover and Loss of Signaling Connection

When a mobile terminal is in contact with the core network (see Figure 1), a physical signalling connection (a dedicated signalling channel) exists between the two bridges. When the terminal moves to another mobility domain, this signalling connection must be released, a new FDBR within the new domain must be contacted, and a new signalling connection must be created. This procedure is referred to as a bridge handover.

However, more than the switching of physical signalling channels is involved in a bridge handover. As already explained, the location register plays a central role in addressing objects with relocatable object references. During a bridge handover, the Location Register is updated and the new FDBR is registered as the current access point of the mobile terminal. This allows future invocation requests to be routed to the mobile terminal through the correct FDBR.

Object invocations that are in progress during a bridge handover, however, present some complications. One of the basic requirements for the mobility bridges is that they must hide the effects of mobility from client and server objects. This implies that the invocations in progress at the time of handover must be reliably and correctly completed despite the momentary break in connectivity that is inherent in a handover.

This reliability is achieved by buffering invocation related messages in the old FDBR until the mobile terminal has successfully connected to a new FDBR. A forwarding connection, termed a tunnel connection, is then set up between the old and new FDBRs and the buffered messages are forwarded to the new FDBR. Because IIOP is a connection-oriented protocol and an invocation reply is always sent over the same connection from which the request arrived, it is not possible to re-route an invocation replies. Therefore, the tunnel connection is be maintained until replies for all pending invocations have been delivered their destination through the old FDBR.

In a wireless environment it is possible to encounter a rapid fade-out that causes a sudden loss of signalling connection before the mobile terminal has the chance to attempt a handover to a base station with a better signal strength.In these cases, it is likely that some messages are lost. Since CORBA communication requires a reliable message service, the bridges must perform recovery after an unexpected loss and subsequent re-establishment of the signalling connection. The LW-IOP protocol provides the means for such a recovery: each LW-IOP message must be acknowledged by the receiver before it can be discarded by the sender. In the event of a communication error, any unacknowledged messages can be re-sent after the communication channel has been re-established. Recovery from a loss of signalling connection also often entails a bridge handover, since the new connection may be established through a different base station.

8. Implementation Experiences

We have implemented the MDBR and FDBR bridges using two commercial CORBA implementations as the basis: COOL ORB r4.1 [1] on the mobile terminal and Orbix 2.2 [3] within the core network. Figure 4 depicts the major software modules in our implementation.

Our initial plan was to implement a request-level bridge as described by the CORBA 2.0 standard. In practise, we encountered difficulties in using the Dynamic Skeleton Interface (DSI) for general purpose bridging. The CORBA standard seems more focused on application-specific bridging, where the semantics of the bridged interfaces are known to the bridge, rather than general purpose bridging. For instance, a separate dynamic skeleton must be registered for each different interface type, causing the ORB to create a proxy object for that interface type. The DSI language mapping also does not provide the means to iterate through the message contents or to manipulate the message contents in a machine independent format. As a result, a general purpose request-level bridge must contain a large amount of platform dependent code and is therefore not portable across different platforms.

Since the option of implementing an inline bridge was not available with commercial CORBA platforms, we decided to implement MDBR and FDBR as gateways that perform the tasks of a CORBA bridge on the Inter ORB Protocol level. Despite the additional burden of writing an IIOP protocol engine, this turned out to be a fortuitous choice. Instead of creating a proxy object in the bridge for each bridged interface, we were able to arrange routing on a message by message basis, only keeping track of the current location of each mobile terminal but not specific objects. The absence of proxy objects simplified the design of a bridge handover considerably, as we did not have to worry about how to rebuild the proxy objects in the new FDBR after a handover. In addition, a large part of the bridge

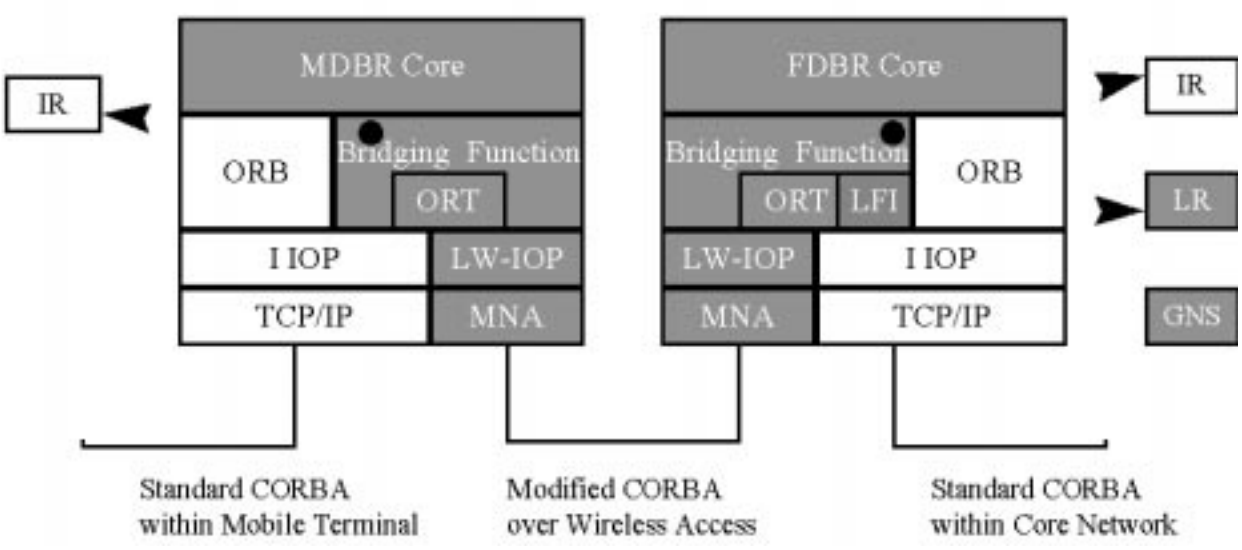

Figure 4. Software Modules in the DOLMEN Mobility Bridges

implementation was common to both MDBR and FDBR. The other considered approaches would have involved less reuse of program code.

Our experiences in implementing a generic bridge lead us to believe that the CORBA 2.0 interoperability standard is not yet mature enough for building general purpose bridges. A generic bridge is required to modify object references that are passed as invocation parameters. This forces the bridge to unmarshal all invocation requests and responses, translate any references, and then remarshal the messages. An external interface repository is required to store the necessary knowledge about each interface. All this amounts to a great amount of overhead that limits the scalability of generic bridges. It is considered that extensions to the IIOP protocol are needed for general purpose bridges to support object reference translation without forcing a bridge to unmarshal and remarshal each message completely. Efficiency is especially important with multi-domain bridging, where a single invocation request might travel through multiple bridges before reaching the server object.

When performing a protocol conversion between IIOP and an environment-specific inter-ORB protocol, access to low-level data representation encoding and decoding operations is essential. These routines exist in some form or other in every ORB implementation, but currently the CORBA standard defines no API that could be used to access these routines.

Finally, the dynamic skeleton interface does not provide access to all features of the underlying inter-ORB protocols. For instance, it is not possible to return a LOCATION_FORWARD status through the DSI to indicate the migration of a dynamic object implementation. The DSI language mapping should also be extended to define a platform independent representation of invocation parameters. The current representation is highly dependent on the hardware platform, operating system, and even on differences between compilers of the same language.

9. Summary

In this paper we have considered CORBA based object communication in the context of mobile networks and terminal mobility. We have shown that it is possible utilise the interoperability mechanisms introduced in the CORBA 2.0 specification to support continuous terminal mobility in object communication. The object interaction model is not affected as the presented mechanism is completely transparent to client and server objects.

Our solution is based on the concept of *interoperability bridges* described in the CORBA 2.0 architecture. We have designed and built a prototype of two such bridges. The *Fixed DPE Bridge* (FDBR) serves as a DPE access point for mobile terminals. The *Mobile DPE Bridge* (MDBR) connects the local ORB domain of a mobile terminal to the core network ORB domain by interacting with an FDBR over the wireless access network. Together, MDBR and FDBR perform location management functions and DPE handovers, enabling terminal mobility on the DPE level.

We have defined a relocatable object reference format that together with the bridges and a special *Location Register* allows us to support relocatable object referencing domains. The location register maintains the current mapping between a relocatable referencing domain and the bridge currently in charge of delivering object invocations to and from that

domain. Bridges update this mapping when a change of location is detected. We have shown that, the mobile terminal being an example of a relocatable referencing domain, this innovative technique can easily be applied to reliable object addressing of mobile objects in a way that is completely transparent to the communicating objects themselves.

We have also enhanced the reliability and performance of object communication in the wireless environment by introducing a *Light-Weight Inter-ORB Protocol* (LW-IOP) between the MDBR and FDBR. The LW-IOP protocol defines efficient message formats and a compressed data representation for object communication. In our experience, LW-IOP significantly reduces the bandwidth consumed over the constrained wireless medium.

Acknowledgements

This paper is based on the work carried out in the EC/ACTS project DOLMEN (AC036) that is a TINA Auxiliary Project. The authors want to thank all their co-workers in the project for fruitful co-operation.

References

[1] Cool orb. http://www.chorus.com/Products/Cool/, 1997.

[2] EC/ACTS Project DOLMEN (AC036). *Definition of an Enhanced Distributed Processing Platform for DOLMEN*, December 1996.

[3] Orbix. http://www.iona.com/Products/Orbix/, 1997.

[4] Object Management Group, New York. *The Common Object Request Broker: Architecture and Specification*, 1996.

[5] S. Trigila. An architecture for integration of fixed and mobile services: The dolmen approach. In *ACTS Workshop on Mobile Multimedia Services*. Signapore, September 1997.