Supporting Disconnected Operation in Mobile CORBA

Niall Lynch
B.Eng.

September 1999

A Dissertation submitted in partial fulfillment of the requirements for the Degree of MSc in Computer Science

University of Dublin
Trinity College Dublin

Declaration

I, the undersigned, declare that this dissertation is entirely my own work, except where otherwise accredited, and that it has not been previously submitted for a degree at this or any other university or institution.

_____

Niall Lynch

September 1999

Permission to Lend and/or Copy

I hereby declare that Trinity College may lend or copy this dissertation upon request.

_____

Niall Lynch

September 1999

Abstract

CORBA has been used successfully for a number of years as a way of building and connecting distributed applications. Normally this has been in the context of a wired network with static hosts. With recent developments in wireless technology, such as GSM and wireless LANs, a new potentially lucrative domain opens itself to the development of distributed applications. As the use of wireless technology becomes more widespread the applications that use it will become more complex. The CORBA architecture could be used to develop applications for this domain. However, CORBA does not yet take into account the problems associated with mobile computing such as limited processing resources on the mobile host and unreliable and low bandwidth wireless links.

The Architecture for Location Independent CORBA Environments (ALICE), went some way to addressing the problems associated with ensuring that CORBA applications could operate successfully within a wireless network. ALICE defines a layered architecture, which takes into account the movement of mobile hosts and ensures that client-server connections remain established transparently to the user.

This dissertation describes how ALICE could be extended to support disconnected operation for applications in a wireless network. The goal was to allow clients residing on a mobile device to continue operation without having to be in contact with remote servers. In this way the unreliability of the wireless link is avoided. This is achieved through the introduction of a new layer in the architecture called the Disconnected IIOP (D/IIOP) layer. The D/IIOP layer provides the additional functionality necessary to allow disconnected operation.

The concept of disconnected operation is not new but is more generally associated with file systems such as Coda. One possible CORBA compliant way of implementing this behaviour was to use Object by Value. Object by Value is part of the CORBA 2.3 standard and enables the passing of an object by value rather than by reference. A prototype of the D/IIOP layer functionality was implemented using the Object by Value approach for moving CORBA objects. This approach was evaluated by enhancing a distributed scheduling application to allow it to work in a disconnected mode by using D/IIOP functionality.

Object by Value did provide a mechanism to move CORBA servant functionality from the server side to the mobile host, and this did allow a client to operate without having to connect to the remote server. This added functionality however, came at a cost of changing both the server and client implementations quite extensively.

Acknowledgements

Many thanks to my supervisor, Dr. Vinny Cahill, for his guidance and advice throughout the course of this project.

Thanks to Mads and Raymond for their willingness to help me with any questions and problems I had during the course of the project

Thanks to other members of the MSc course for their friendship and for helping make the year a lot more interesting.

Special thanks to Caroline for all her help over the year and especially for answering all my silly questions.

Finally, thanks to my family whose constant support throughout the year was much appreciated.

Table of Contents

Table of Figures

Chapter 1

1. Introduction

Advances in wireless networking technology have resulted in completely new ways of communication and computing. Users can carry portable devices and have access to a shared infrastructure independent of their location that will allow them to use numerous useful applications. In the resulting environment users can communicate with each other easily and there is continuous access to services provided by the wireless network. Two domains, the mobile telecommunications domain and the mobile computing domain are driving this scenario. Mobile voice telephony is obviously a major commercial success story with the result that the mobile operators are moving towards providing data services as well as standard voice communication over their networks. Mobile computing is also evolving to offer more applications on smaller devices that are continually connected to the network and can use networked services. The applications that are available on the mobile device and the types of scenarios that they can be used in will be a determining factor in how successful mobile computing will become. This chapter looks closely at the evolution of mobile telecommunications and wireless mobile computing and how they have become of huge commercial interest, i.e. the business case for investigating this domain. A description of the sort of applications that are being deployed in existing networks is also provided giving an indication of how these applications are expected to evolve. The chapter then looks at how the Object Management Group's (OMG) [OMG ‘98] Common Object Request Broker Architecture (CORBA) approach to building distributed applications can have a role to play in the development of applications for mobile devices. Some aspects of the CORBA model need some modifications if it is to be used in the wireless mobile environment. This chapter introduces the Architecture for Location Independence in a CORBA Environment (ALICE) that allows CORBA to be used in the wireless environment. The concept of disconnected operation and how it could be integrated with ALICE to enhance the use of CORBA applications in a mobile environment, which is the primary focus of this thesis, is also introduced. Finally a roadmap of the rest of the dissertation is presented.

1.1 Telecommunications Mobility

The mobile phone has been a huge commercial success story that continues to get better for the mobile telephony operators. Europe's top twenty cellular operators are worth more than $400 billion. The number of customers within each European country is growing at a phenomenal rate. In Finland the number of mobile phones has outnumbered the number of fixed line phones and continues to rise. Telecom Italia Mobile is Europe's largest mobile phone operator with 15 million subscribers. It is expected that by the year 2000 that there will be over 500 million mobile users worldwide. This growth has been solely based on providing voice telephony, which, not surprisingly, has proven it to be the killer application in getting people to use a mobile phone. GSM [Redl] is the predominant digital mobile telephony standard in use and is referred to as a second-generation mobile standard. The first generation covered the analog mobile networks while the third generation is what is known as Universal Mobile Telecommunications System (UMTS). The following subsections look at the evolution of GSM, the services that will be provided using GSM and the technologies facilitating those services, with the ultimate goal of progressing to the UMTS vision.

The telecommunications companies are now searching for more revenue generating ways of leveraging the mobile phone. To date mobile phones have been extremely successful in offering a voice service in the same way as the fixed line phone. In order that further channels for revenue are found and a better product is offered to their customers, mobile phone operators have set out to offer value added services on the handset such as calling line ID. They have also been looking at offering more intelligent services such as database lookup of calling line ID to provide the full details of the caller to the callee. These types of services are also available on fixed line phones so mobile phone users would expect to have them. A major effort is also being made in the area of enabling data communications over the mobile network.

Data communications has a big part to play in the development of what are known as smart phones and the services they will offer to customers. Smart phones are phones that offer data applications to the user in addition to the standard voice, calling line ID and address book services that are standard on most phones. At present GSM networks do support one data service called Short Message Service (SMS). SMS is limited to carrying 256 characters per message, but nevertheless is being used to exchange emails. However, the fact that SMS has taken off commercially in some countries points to the potential of data services. For example in Finland mobile data applications now account for nearly 10% of the revenue for Finland's cellular operators, and up to a third of their profits. The data traffic is highly profitable given

that the cost of carrying messages is very low [FT '99]. There are SMS based information services on offer covering areas such as news, sport, finance, stock market information, currency rates and job vacancies.

The next step for many telecommunications companies appears to be enabling access to the Internet in some form or another on their phones. This would allow customers to send emails (using standard mail protocols rather than through SMS) and browse the web. They also would like to offer proximity services i.e. services based on the user's location. E-commerce also presents a lucrative domain where the mobile phone could potentially become the ATM in the pocket for example, users could download cash onto a card through their mobile phone. Another possibility is in streaming media services such as audio and video. These could be extended to offer video-conferencing services.

The following diagram illustrates how the wireless networks are adapting to facilitate these changes, moving from providing reliable voice communication to providing data services on the mobile device. The starting point for the evolution is the second-generation GSM network and the goal is the third generation UMTS network where data transmission rates should be 2Mbit/s. On that path there are a number of proposed technologies that should increase the data rate.

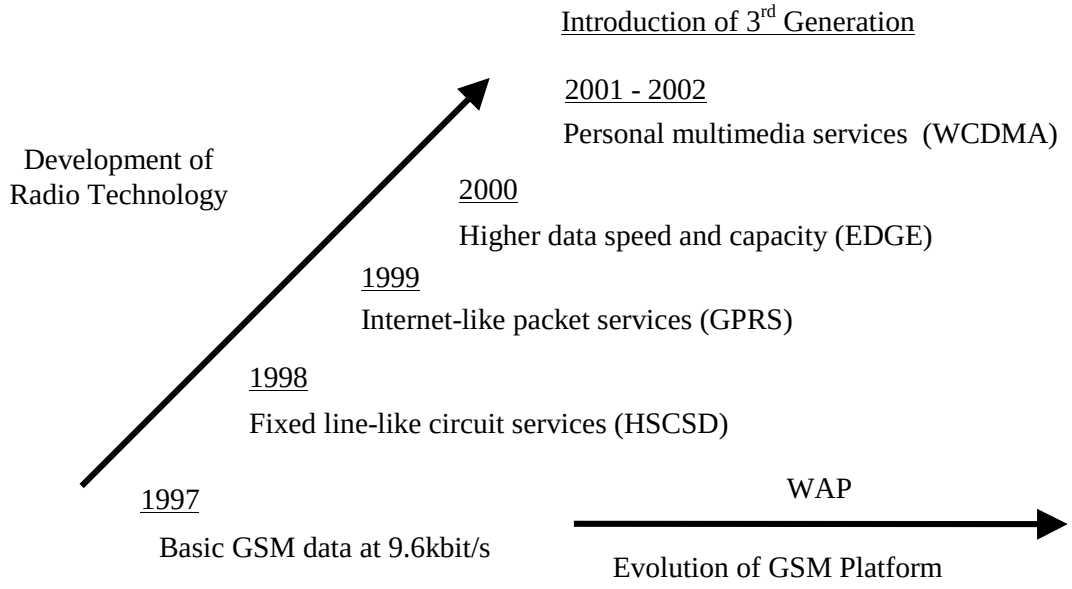

Figure 1-1 Evolution of Telecommunications Wireless Networks

The major limitation on access to the Internet from mobile phones at present is bandwidth. GSM offers 9.6kbit/s as a data rate. However, a lot of investment is going into new technologies that will increase bandwidth and facilitate easier and faster data communications

in wireless telephony communications networks. The following sections describe some of those initiatives relating to provision of data services that are in the path from GSM to third generation mobile telephony.

HSCSD **-** High Speed Circuit Switched Data represents an initiative by the telecommunications operators to achieve greater throughput to the mobile device thereby providing the capacity to enable more data applications. HSCSD is connection-oriented meaning that the two end points of the communication are connected, like a telephone call. This means that it is more suited for applications that require low latency, such as mobile video-conferencing. GSM uses Time Division Multiple Access (TDMA) to divide each channel into eight time slots. Each user is allocated one transmit and one receive slot with a transmission rate of 9.6kbit/s. 14.4kbit/s is now available by reducing the redundancy in the data. The move from 9.6 to 14.4kbit/s requires only small changes to the terminals and the infrastructure. HSCSD uses multiple TDMA slots, allowing for Nx14.4kbit/s. Operators are indicating that they will start with a two-slot (28.8kbit/s) service and then offer an asymmetric combination of four slots for Internet access. This will give a downstream bandwidth of 43.2kbit/s and an upstream bandwidth of 14.4kbit/s. New handsets are required for HSCSD and these are likely to accommodate a maximum of four slots. This means that it is likely that bandwidth for HSCSD will top out at 43.2kbit/s, however using data compression can increase this bandwidth [Emmerson '98].

GPRS **–** General Packet Radio Service is another protocol that is intended to improve data communications over wireless networks. It is a packet-based method of communication over the GSM infrastructure. Since it is packet based it is suitable for bursty data applications such as web browsing and is ideal for TCP/IP environments. It is basically a GSM overlay network and changes to handsets will be needed to accommodate this new technology. A big impact of using a packet-switched protocol will be the need to change the business model. Customers will no longer be charged based on a unit of time but they will be charged 'per bit'. There are four different coding schemes, which results in four different single channel data rates ranging from 9.05 up to 21.4kbit/s per channel. A GPRS channel is equivalent to a HSCSD time slot. Up to eight channels can be used for sending data, giving a maximum bandwidth of 171.2kbit/s.

GPRS will be introduced by some operators towards the end of 1999 and 2000, although some operators may wait for UMTS. However, in the corporate world it will prove popular since it

reduces costs and increases communication since costs are based on volume not on time and distance.

Nokia has plans to implement both HSCSD and GPRS in future versions of its mobile phone. This will allow a subscriber to choose between the two services depending on the type of application they are using. [Silber '99]

Wireless Application Protocol **-** As data rates in mobile communications increases, it is important to look at the higher application level that should leverage the extra bandwidth to provide applications. There will be a demand for nicer email tools and possibly World Wide Web browsers on the mobile terminal. People are used to browsing from their office and they would like to continue that while away from the office. A lot of office workers spend a lot of time away from their offices, even when they are present at work. There are obviously great benefits for workers being able to access their corporate Intranet. At meetings or while they're away from their office they can get the latest organisational information, they can launch spreadsheets and graphs, they can get information about people and any other information they would normally use on the office desktop. Outside of the corporate domain there are also many opportunities for mobile device services related to browsing the Internet. For example informational services like news and weather and services relating to location.

A problem facing the telecommunications companies at the moment is how to offer reliable application services to the mobile end users. This means having the required bandwidth to run the application, ensuring that the application continues to run as the user roams and the connection may become more unreliable. Another problem is what type of data application the user will want to use on their mobile phone. At present there are not many applications that could run on the mobile phone apart from some informational services that users could scroll across the display of the phone. Telephone companies are now developing new types of phones with larger displays that offer more interactive services to the users such as Web browsing. A protocol that was developed with this as an aim was the Wireless Application Protocol (WAP).

The WAP Specification aims to bring the telecommunications industry a step closer to the mobile Internet defining an architecture for the delivery of Internet content to wireless devices. It was developed by the WAP Forum, which is a group of organisations, whose goal is bringing Internet content and advanced services to mobile devices. It essentially allows the handheld device decide how to display the information available at the server. The organisations responsible for setting the Forum up were Ericsson, Motorola, Nokia and Unwired Planet.

However, it is obvious that standard browsers will not function on a mobile device as it exists today, or even in the near future, firstly because of the display constraints and secondly because of the limited bandwidth available for mobile data communications.

With WAP the low bandwidth and device limitations are taken into account. The philosophy is to utilise as few resources as possible on the handheld device and compensate for its limitations by enriching the functionality of the network [Parrish '98]. The content is delivered from standard Web servers. The content can be authored in HTML or directly formatted in the Wireless Markup Language (WML). A scripting language called WMLScript has also been defined. These languages assume that the normal keyboard and mouse inputs are not available to the user and that the display is limited. A microbrowser has also been defined for the wireless terminal that defines how WML and WMLScript should be interpreted in the handset and presented to the user.

A WAP Proxy exists which acts as a gateway between the standard Internet protocols and those specified by WAP. So the proxy takes the information requested by the user and packages it for download to the device according to that devices characteristics, thus it optimises use of the available bandwidth by only retrieving information that can be displayed.

It is important to note that the WAP standards were developed so that they complement existing standards. The standard doesn't specify how data should be transmitted over the air interface. The protocol is intended to sit on top of bearer channel standards like the ones mentioned earlier, GSM (SMS), HSCSD and GPRS. The two standards working together will provide the complete end user product solution.

The following diagram illustrates the main components of the WAP architecture [UP '98]. The gateway proxy resides on the fixed network and acts as the access point to the content server and prepares the content for presentation on the wireless device. The gateway takes over all DNS services to resolve domain names used in URLs, removing this requirement from the handset. It translates WAP protocol requests to HTTP and TCP/IP requests.

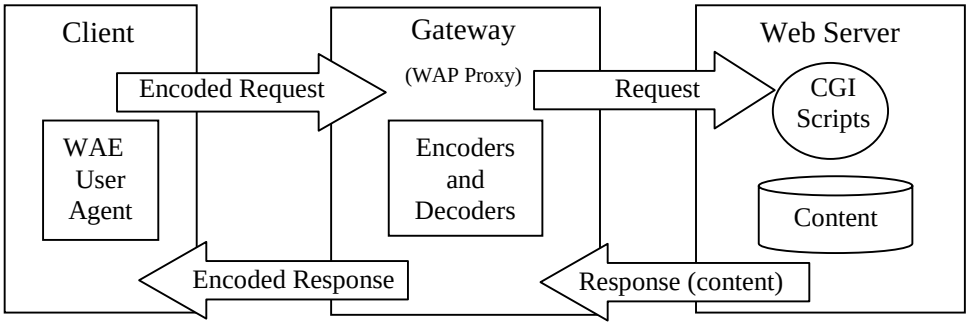

Figure 1-2 The WAP Programming Model

A protocol stack has also been defined that ensures that a wide variety of network types can run WAP applications. It makes a number of enhancements to make HTTP more suited to the wireless network environment. For example the plaintext headers of HTTP are translated into binary code that significantly reduces the amount of data that needs to be transmitted over the air interface. A lightweight session re-establishment protocol has been defined that allows sessions to be suspended and resumed without the overhead of the initial establishment again. This allows the suspension of a session, which means saving battery power or freeing up the network resources.

WAP is a comprehensive protocol that brings a lot of good ideas to the mobile telecommunications domain. It takes into account the desire of companies to offer more data services on mobile devices and accepts that there will be no standard mobile device and so it is adaptable. It also takes into account that the access protocol for the mobile device could be one of many possibilities and therefore is not tied to any. Combined with the probability of increased bandwidth to mobile devices and the improvement of the mobile devices in their ability to display information, WAP provides a real solution for providing data applications on mobile devices in the near future. In fact phones that support WAP will be available on the market by the end of 1999.

UMTS **-** UMTS the third generation mobile telephony in Europe is known as International Mobile Telephony (IMT 2000) globally. The goal of UMTS is to have an integrated approach to offering voice, data, fax and paging facilities enabled through technologies providing higher bandwidth. One of these technologies is Wideband Code Division Multiple Access (W-CDMA). Europe and Japan will start offering 3rd generation services by the end of 2001. For the 3rd generation network the goal is to standardise on one technology worldwide by the year 2000. IMT 2000 will provide a data rate of up to 2Mbit/s. The third generation network will

herald the convergence of mobile phones and the Internet. With high-speed wireless data access and intelligent networks the future will see a more application rich computing scenario for the mobile telecommunications devices where they will play a larger role in how people work, shop, pay bills, keep appointments, keep track of the children and entertain themselves. It is reported that UMTS will be worth Euro 10 Billion per year in Western Europe by 2005.

With this sort of bandwidth available to the mobile device and the fact that the UMTS mobile device will provide integrated services it is obvious that the devices are no longer just phones but small computing devices, the smart phones that the telecommunications organisations have been developing as bandwidth grew. The types of services that could be offered on these devices will be much less limited than they are today.

1.1.1 The Future of Mobile Telecommunications

It appears that over the next few years a big effort is being made in the mobile telecommunications world to improve the existing services offered to mobile customers and also increase the number of innovative services, this means the introduction of many data services. This will be facilitated by the progression from the 2nd generation network to the 3rd generation network. The general process through which services are developed will take on a more computing like approach rather than a traditional telecommunications approach. The various tools at disposal of the computing world will be used by the telecommunications world to offer new and innovative services. It is obvious that there will be a convergence of what the telecommunications world and the computing world will offer in the area of wireless data communications, with both sides attempting to provide innovative services that are suited to mobility.

The following section looks at how the computing world addresses the issue of mobility and how some aspects of the development of mobile computing are converging to the same point as that of mobile telecommunications.

1.2 Mobility in Computing

The most visible form of mobility in the computing world has been the portable computer or laptop. The laptop essentially provides the same applications a user can expect to have on their desktop computer in the office. The laptop has proven to be an essential tool for the mobile worker meaning computing capabilities are no longer restricted to the office. The way in which people conduct their work has become a lot more flexible. Laptops have evolved over the past

number of years to become smaller lighter devices that are much more convenient to carry around. The goal of providing smaller lighter more portable and convenient computing devices has also lead to the development of Personal Digital Assistants (PDAs). These devices offer a restricted set of applications to the user such as an address book and personal organiser, but they are very small and light and easy to use. PDAs are gaining in popularity as they continue to remain small but offer an increasing range of applications to the user. They can also be connected to the users desktop computer through a serial port and files can be exchanged between them. They can be seen as a portable arm of the computer that allows the user to continue to use certain applications while they are away from their office desktop computer. They differ from the laptops in that they do not try to support the same range of applications that the office computer can support.

In a highly competitive world demands being placed on employees are becoming greater. This includes being able continue working regardless of location. This has lead to the desire, regardless of location to be able to connect to the users local network, where the services they are familiar with reside. One possible way of supporting this is through a wireless interface. Ideally, users of wireless networks will want the same services and capabilities that they have commonly come to expect with wired networks. The next section points out some of the problems faced in wireless mobile computing and the following sections describe some of the technology trends aiming to address those problems.

1.2.1 Issues in Mobile Computing

Wireless communication is much more difficult to achieve that wired communication. There are many factors that may affect a wireless connection. The wireless connection is open to the influence of the surrounding environment. This may affect the signal producing noise or blocking the path.

Interference and Reliability **-** Interference in wireless communication occurs when there are signal collisions i.e. when there are two or more sources sharing the same frequency. This can happen when multiple stations waiting for the channel to become idle then begin transmitting at the same time. Collisions can also be caused by the "hidden terminal" problem when a mobile terminal begins transmitting without detecting that there is a transmission already in progress.

Mobility and Location Management **-** This is the major advantage of a wireless system over a fixed network. Users remain connected while they are changing location. This also places the

demand on the wireless system to have some mechanism to accommodate handoff across transmission boundaries and to route traffic to mobile users. The services available to a mobile terminal may change depending on its location. For example, access to a particular service may no longer be available if the terminal has moved to a new domain by crossing an international border.

Bandwidth and Latency **-** Performance over wireless links is limited by low bandwidth and high latency. Ideally wireless networks should have a bandwidth approaching that of their wired network counterparts. Performance of transport protocols, such as TCP and NFS, is much lower since these protocols were designed for use in fixed networks. For example, in a wireless network TCP incorrectly interprets packet loss as a sign of congestion in the network and immediately throttles packet transmissions. This causes unnecessary performance degradation and loss of throughput. NFS responds to losses with idle times of at least a number of seconds. These type of recovery mechanisms limit the performance of wireless networks. Modifications to these protocols to adapt to the wireless network characteristics would improve performance and also maintain compatibility.

Disconnection **-** There are times when the mobile host may be out of range and find itself disconnected from the network. This renders the mobile terminal unable to avail of any of the network services normally available to it. Disconnection form the network could also be a user's choice since the cost of maintaining a connection for an extended period of time may be prohibitive.

1.2.2 Technologies and Trends

There are a number of technologies and approaches that the computing world has adopted in order to deal with the issues presented in the previous section. The following list is not exhaustive but is intended to illustrate some of the major technological developments.

Wireless Local Area Networks - IEEE 802.11- Wireless Local Area Networks (WLANs), like their wired counterparts, are being developed to provide high bandwidth to users in a limited geographical area [Crow '97]. WLANs are being used as an alternative to wired LANs since they do not have the high installation and maintenance costs associated with wired LAN infrastructures. The ITU-T 802.11 specification deals with all aspects of WLAN technology. The purpose of the standard was to provide wireless connectivity to equipment that required rapid deployment and that might be portable within a local area. The mandatory transmission rate is 1Mb/s with optional support for a 2Mb/s data transmission rate.

In dealing with the issue of mobility they are more reliable than cellular networks; it is easier to ensure a good wireless connection if the terminal is within a local area than if it is hurtling down the motorway. High-speed handoffs are not an issue. Location is obviously restricted to the local area and WLANs do not support nation-wide roaming. But there are many examples of networks whose applications do not require nation-wide roaming; campus networks, networks for mobile workforces e.g. in a car assembly plant where information can be passed between devices used by workers, in hospitals where staff can access latest information. Even networks for police forces could be deployed, as the technology is developed to cover wider metropolitan areas [Phifer '98].

In terms of bandwidth most of the commercial WLAN products provide data transmission rates of 2Mb/s, however, this is significantly slower than the standard 10 Mb/s Ethernet LANs offer. Standards and technology are being developed to boost transmission rates of wireless LANs from the 2Mb/s regime to tens of megabits per second. There are research prototypes of wireless LANs based on asynchronous transfer mode (ATM) technology running at 10Mb/s and serving as testbeds for mobile networking at much higher rates. If wireless networks are to be a truly viable technology they will need to transfer data at the rate of 10Mb/s [Regan '97] [Alkh '97]. This is a major growth area and within the US it is believed that by 2003 WLAN shipments will grow to $2.6 billion [FT '99].

Ad Hoc Networks **-** These types of networks are used when the existing communication infrastructure is expensive or inconvenient to use. In an ad hoc network, each mobile node operates not only as a host but also as a router, forwarding packets for other mobile nodes in the network that may not be within direct wireless transmission range of each other. Mobile nodes dynamically establish routing among themselves to form their own networks "on the fly". Ad hoc networks approach the issue of mobility by making the mobile node adaptable to its new location in that it is able to configure a new local network with other nodes in its new location and continue to offer services to the user available in the new local network.

Bluetooth is one technology that is making a major impact in ad hoc networks. Bluetooth provides the transport capability for an ad hoc network through its radio interface in the unlicensed Industrial Scientific Medical (ISM) 2.45GHz frequency band. It enables mobile devices to connect and communicate over short-range wireless links in ad hoc networks. It resulted from a research at Ericsson investigating the feasibility of a low power and low cost radio interface between mobile phones and other devices. The air interface has been optimised to provide maximum immunity against interference in the 2.45GHz band, which it shares with

wireless LANs and microwave ovens. The specification originated from Ericsson has gained support from Nokia, IBM, Toshiba, Intel and many other manufacturers. It is a combination of software, communications protocols and a tiny radio transceiver on a chip that will let devices communicate (both voice and data transmission) with one another over distances up to 10 metres. The range can be increased to distances of 100 metres with the addition of an optional amplifier. Devices with Bluetooth chips in them will begin to appear on the market at the end of 1999. Examples of such devices are mobile phones, modems, headsets, PDAs, PCs, projectors, local area networks and so on. It paves the way for new and completely different devices and applications [Haar '98]. For example a laptop user enters a building where they have never been before but would like to print a document. If both the laptop and printer were bluetooth enabled they could communicate over that radio interface and create an ad hoc network. A document to be printed could be passed over this network from the laptop to the printer. In terms of bandwidth Bluetooth devices will communicate at up to 720 kb/s with a possibility of higher rates depending on the number of nodes in the ad hoc network.

PDAs - PDAs are evolving and many are starting to incorporate a wireless interface. With a wireless interface and the possibility of wireless data access the number of services they offer will grow to include access to the Internet. They will also have voice applications placing the computing companies in competition with the smartphone products to come from the mobile telecommunication operators. In fact the Palm VII connected organiser from 3COM is already on sale in the US and it provides wireless access to the Internet. To deal with the bandwidth limitations imposed on using their wireless device Palm Computing® came up with the concept of "web-clipping" for their Internet access application. Web clipping is based on the idea that less data transferred results in a more efficient system. To do this they use two principles, firstly all user interactions are based on a simple query and response rather than on a system of hyperlinks and secondly application partitioning is used whereby the query portion is stored locally on the handheld. With the query portion stored locally the user enters the request in a form before even going on line, then the user submits the request and the resultant page or web clipping is returned, which is very small. On a typical application the query is about 50 bytes and the returned page is less than 500 bytes. This approach accepts that the bandwidth will be restrictive but by changing how the application operation it is possible to provide acceptable performance to the user. The Web Clipping Proxy server is responsible for converting the standard Internet protocols and content from the web page into a form that's tuned for transmission across a wireless network for display on a small device. The Web Clipping Proxy server implements a reliable layer over the UDP protocol to talk to the Palm

VII handheld. Its use enables one packet to be sent up as a request and one or more to be sent downstream containing the web clipping. This protocol greatly reduces latency and conserves battery power relative to using TCP [Palm '99]. This initiative is very similar in design and approach to the WAP forum work. The Palm VII is only available in the US through a deal with BellSouth Wireless Data, whose network covers over 260 of the most populated areas in the US. It is interesting to note that this is a dedicated wireless network and not a voice network carrying data. Up to now the approach for using exchanging data over wireless networks has been by using a modem from the laptop or the handheld computer connected to a mobile phone, which could enable the sending of data.

Another initiative in sending data over wireless links is through the use of the protocol Cellular Digital Packet Data (CDPD). CDPD is also known as wireless IP since each device is given an Internet IP address users can use any TCP/IP-based application such as News, Telnet, Ping and so on. This means IP goes over the wireless link, unlike the situation with the Palm VII connected organiser where a proxy is used. CDPD is used within the US D-AMPS (equivalent to GSM in Europe) mobile telecommunications network to carry packetised data (IP packets) between the mobile terminal and the mobile base station at 19.2Kb/s. Packets are then routed across the Internet or other IP networks to the destination. By using datagram packets, CDPD-based applications can better adjust to packet loss and delay, problems that arise when using wireless links.

Convergence - Both the mobile computing domain and the mobile telecommunications domain have evolved over the past number of years to a stage where the two domains are beginning to converge by sharing similar goals. A major shared goal is to overcome the limitations of the wireless connection such as low bandwidth and high latency in order to provide valuable applications to the users of the portable handheld devices, which are either smart phones or handheld/mobile computers. This goal has seen the development of protocols like GPRS that increase the bandwidth in wireless telecommunications networks such as GSM and protocols such as WAP at the application level and CDPD at the network level that aim to bring the Internet to the handheld. One consortium that envisaged a convergence of the mobile telephony world and the mobile computing world was Symbian. Symbian has developed an operating system for both smartphones and PDAs called EPOC [Comer '98]. These developments highlight the effort being made to enhance the services offered to mobile device users. The following section illustrates some of the novel applications that are being deployed now and

how applications will be the driving force to the success of mobile wireless communications/computing.

1.3 Applications are the Future

Applications in the future will determine which mobile vendors will succeed. The types of services will be the differentiating factor. Lots of research is ongoing into many fields where mobility will make people's life easier and more efficient. It will fundamentally change how people think about the work place, no longer will people be tied to a location. There are many commercial information services, which could be offered to mobile subscribers. Mobile devices will become more intelligent and may even make decisions based on location and user profiles.

Telecommunications Services **-**. Operators are having trials for other non-voice applications, which include the customer activating the service by dialling a particular phone number. There is a dial-a-drink service where the customer can buy a drink from a vending machine by calling a free phone number displayed on the machine. The machine contains a GSM phone, which activates the dispensing mechanism. The cost of the drink is charged to the caller's phone bill.

Phone companies are moving away from just being bit pipe providers and into the higher margin areas of end user applications. Some phone companies are putting GPS chips into their phones so the users location can be pinpointed and services offered related to that position. Another major set of services to be offered on future phones will be Internet services such as web access and email. By using a technology such as WAP mobile phones will be capable of offering a web browsing facility and an email facility.

Corporate Computing **-** To fund market growth it is necessary to have corporate acceptance of wireless computing. To enable this wireless will have to be successfully integrated into the corporate network. Wireless remote access must be secure and manageable. Wireless devices have to offer seamless access to the network operating system and Intranet servers and databases in a way that doesn't upset the corporate backbone.

At present there are a number of enterprises taking on board wireless communication and this is causing them to re-engineering business operations to create a world without cables.

On such industry adopting this technology, which isn't immediately obvious, is based on the ski slopes of Colorado. In Vail Resorts they have deployed a Proxim Wireless LAN which covers some of the 50-mile area of ski slopes. The wireless LAN is based in the ITU-T 802.11

specification. Changes in schedules for instructors were difficult to communicate especially if the instructor was already out on the slopes. To deal with the problem they needed to be able to track in real time which guest was skiing with which instructor, where they were skiing and who cancelled and who re-scheduled. With the wireless LAN installed this information is easily available at all times to instructors and supervisors. A wireless Casio Casiopeia E-11 Palm-size PC is used and Proxim provided the wireless LAN equipment. The palms PCs each have a Proxim RangeLAN2 wireless PC adapter card and the necessary drivers that enable remote communication. There are six base facilities and at least two access points for each. The project is such a success there is a plan to add more to improve coverage to more parts of the resort.

Campus Networks - Colleges are now viewing wireless technology as a cost-effective way of deploying more accessible computing facilities for their students.

At the University of Oklahoma's College of Engineering (COE), the students don't just use the computers to surf the web. They submit assignments via email and participate in course related chat discussions. They can also download video coverage of a missed lecture. However, the unique thing is that they can do this from any location on campus, they don't need to queue to get access to the network. They use a Proxim wireless LAN and each student has been given a Laptop with a Proxim RangeLAN2 card for wireless network access. There are more than 700 students and staff are connected. They have also adapters for the stationary PCs so that they can access the wireless network too. The only problem is bandwidth, with rates being quite low meaning that the network is slow for students accessing and uploading data. However, with the new generation wireless LAN products coming on line this should be solved.

The Medical World - One area where mobile computing could obviously be deployed to provide a better service is in the medical profession, and particularly in relation to patient records. It would be much more efficient if doctors and nurses did not have to carry around and file paper records as they deal with patients in a hospital. MacNeal HealthCare centre in Chicago decided to install a wireless LAN at one of their hospitals. They used wireless LAN products from BreezeCOM called BreezeNet Pro, which are a line of plug and play wireless Ethernet products including access points, station adapters, PC card adapters and Ethernet bridges. The portable devices were pen based handheld computers. Again the standard used was the ITU-T 802.11 wireless LAN standard. Again, bandwidth was the only significant problem meaning the network appeared slow for the users.

In the Home - The ad hoc network technologies are being geared towards facilitating services in the home environment. Bluetooth and similar proposals such as Piano from Motorola see a world where all electronic devices have a wireless interface. These devices can then communicate across their ad hoc network to other devices in their vicinity. The ad hoc network infrastructure enables the devices to communicate but there are no services as such deployed that uses this capability. Possible services in the home include a monitoring system for heating and lighting and general energy conservation. There is also scope for use with the security system for the house. This could also be linked to the Internet so the user could get an email or a SMS message if there was some breach of the system [Pickar '99]

Wireless devices are being used in an increasing number of domains. The applications are becoming increasingly complex and many have a distributed nature. This is where CORBA could be introduced as an approach for developing distributed applications for the wireless domain. The next section introduces CORBA and how the OMG has realised that it must change if it is to be used in a wireless environment.

1.4 CORBA

CORBA defines an object-oriented framework for developing distributed applications. This framework makes network programming easier by enabling the development of distributed applications as if they were being implemented for a single computer. CORBA defines a standard architecture for Object Request Brokers (ORBs). An ORB allows the creation of server objects whose member functions can be invoked by client programs anywhere in the network. The General Inter-ORB (GIOP) was defined to enable interoperation between ORBs and a mapping of this onto TCP/IP is called the Internet Inter-ORB Protocol (IIOP) and has become the de-facto standard for ORB communication.

Realising the potential of wireless computing, how mobile devices were becoming increasingly more powerful and sophisticated and that users would expect to access the services they normally access from stationary devices from their mobile devices, the OMG issued a Request for Proposals (RFP) on wireless access and terminal mobility. The RFP was issued in May 1999 and looked for proposals on technology that would allow mobile terminals to host both CORBA clients and CORBA servers. The RFP pointed out that with wireless networks the client-server interaction is affected and that interaction mechanisms for fixed environments need to be changed to suit this environment. The RFP is based on issues presented in a white paper from the OMG on wireless access and terminal mobility. It identified two aspects of the CORBA specification that required changes and extensions to allow CORBA to exist within

the mobile computing paradigm, the application level and the protocol-level. At the protocol level investigation should be about what needs to be done to allow IIOP, the lingua franca of CORBA, to be used in the problematic wireless environment. At the application-level mechanisms need to be found that allow applications to intelligently deal with wireless network problems such as disconnection/reconnection high latencies and fluctuating bandwidth. This thesis looks at the application level issues and how disconnected operation may provide a mechanism that helps to hide the inherent problems of the wireless network.

1.5 ALICE and the Project Goal

ALICE allows CORBA objects running on mobile devices to interact transparently with objects hosted by off-the-shelf CORBA implementations. In the ALICE architecture a mobile host communicates with a mobility gateway over a wireless link. The mobility gateway acts as a proxy for the mobile host and relays communication to and from the mobile host over the fixed network. The ALICE framework defines a layered architecture that allows its IIOP implementation to continue normal operation even if there are times when the host connection to the mobility gateway is down or when the host is roaming and the mobility gateway that it uses changes.

The primary goal of the project was to enhance the ALICE framework so that client applications that used ALICE could continue operation in cases where the link between the mobile host and the mobility gateway was down for an extended period of time. Distributed object computing provides mechanisms for the replication and migration of objects. Many projects have used these mechanisms to provide a way on bypassing the problems inherent in using a wireless link. One such mechanism is disconnected operation. This is where the server side application code is replicated on the mobile host, client side. It was intended to define a new layer in the ALICE protocol stack called the Disconnected IIOP D/IIOP layer that would provide the necessary mechanisms to support disconnected operation. This would enable the movement of CORBA objects from the server side to the client side, where the client could invoke on the server objects without using the wireless link. The concept of disconnected operation has been used in file systems such as Coda, but applying it to the CORBA domain is relatively new.

Another goal of the project was to use a standard CORBA approach to support the mechanism of disconnected operation. One possible way was with Objects-By-Value which is part of the CORBA 2.3 specification. It was intended to evaluate this as a mechanism for implementing the D/IIOP layer functionality. To evaluate disconnected operation it was decided to implement

a distributed scheduler application that would use the Objects-By-Value specification so that its server side functionality could be moved to the client side enabling disconnected operation.

1.6 Roadmap

The remaining chapters in this thesis catalogue the phases of the project that were carried out in order to achieve the ultimate goal mentioned above. The following is an outline of the chapters:

Chapter 2 CORBA and Mobility

This chapter presents some of the problems associated with computing in a mobile environment and introduces some distributed computing techniques to overcome these problems. It also presents how a number of research projects used those techniques to provide mobile computing architectures. It then moves onto look at how CORBA can operate in a mobile environment and how ALICE addressed some of the associate issues. Finally it introduces the concept of disconnected operation.

Chapter 3 Extension of ALICE - D/IIOP

This chapter presents the proposed design and enhancement to ALICE that is required to support disconnected operation. It also suggests some possible approaches to supporting this functionality.

Chapter 4 Implementation

This chapter presents the implementation issues and how Object by Value was used to support disconnected operation. It also shows how a distributed scheduling application was enhanced to allow it to work in a disconnected mode.

Chapter 5 Evaluation

This chapter evaluates the use of Object by Value for supporting disconnected operation and also looks at the bigger question of using disconnection as an approach to bypass the problems of a wireless link.

Chapter 6 Conclusion

The conclusions along with some proposed future work are given in this chapter.

Chapter 2

2. CORBA and Mobility

Wireless communication is much more difficult to achieve than wired communication because the surrounding environment interacts with the signal, blocking signal paths and introducing noise and echoes. As a result wireless connections are of lower quality than wired connections with lower bandwidths, higher error rates and more frequent spurious disconnections. These factors can in turn increase latency due to retransmissions, retransmission timeout delays and error control protocol processing. Wireless connections can also be lost and degraded due to mobility, if the mobile terminal goes out of range or if some obstacle is in the way of the signal. The number of devices in a cell can also determine whether a mobile terminal performs well, if there is a high concentration of terminals, the network may become overloaded. With the proliferation of mobile devices and various mobile networks another problem that will arise is the interworking of various mobile devices from different networks. Users do not want a situation where different mobile devices are required for different mobile usage scenarios.

This chapter looks at some issues related to mobility and discusses how distributed object architectures can be used for mobile computing giving some examples of the mobility aware architectures developed in research projects. This is followed by an analysis of using CORBA for mobile computing and the OMG White Paper and Request for Proposals (RFP) on issues relating to using CORBA in mobile computing. Next there is a description of ALICE and how it goes some of the way to answering questions in the RFP. Finally disconnected operation is presented as a way of extending support for mobile applications with reference to a number of projects that use this approach.

2.1 Mobile Computing Architectures

Using Distributed Object Technology to support mobile computing brings with it a number of mechanisms that help overcome the problems inherent in this domain mentioned above. One of the advantages of the distributed object paradigm is that it can deal with heterogeneous environments where there are multiple platforms and operating systems by providing a modular and abstract way of representing these environments. As mobile computing becomes more widespread this will become a major issue especially as there will be diversity in devices and standards such as Bluetooth, Wireless LANs, GSM and UMTS. Users will want to use the same devices across these different platforms. Furthermore, with the rise in the number of

users extra demands will be placed on mobile services. It has already been illustrated that bandwidth and latency of the wireless link can affect how applications operate on a mobile device. This coupled with the mobility of the user, giving rise to situations where disconnections can occur mean that wireless link presents the core source of problems with mobile computing. By using techniques of object replication, migration and delegation availability can be improved. [Chen'97]

Before discussing these techniques it is useful to describe a general architecture for mobile communications to which they could be applied:

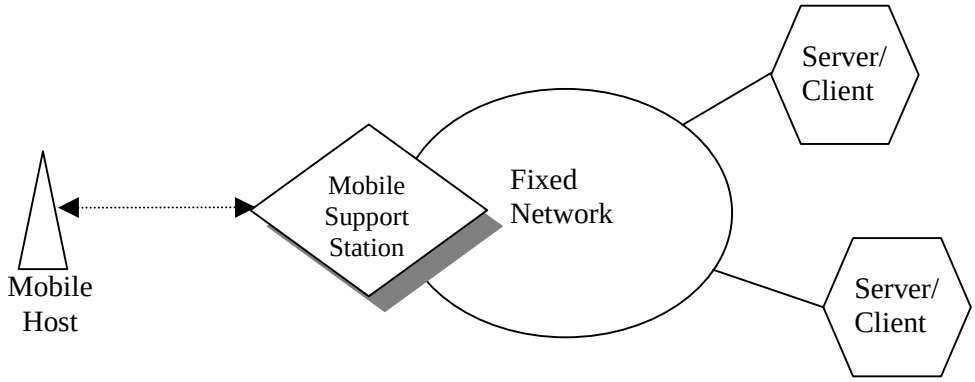

Figure 2-1 Mobile host using services of the fixed network

A typical scenario would involve server objects providing services such as shared data storage, information retrieval, or application components. The objects may be classified application objects that provide application or service functionality, and data objects that provide storage and retrieval. Client applications running on mobile hosts can access shared resources provided in server objects. The server objects may also exist on mobile hosts and they may also be able to migrate or replicate to secondary hosts such as the mobile host. The mobile support station is a fixed host that contains hardware and software support for mobility, including wireless network interfaces.

Object Replication **-** To provide increased availability of server resources, server objects may be replicated and placed at different nodes within the network and on mobile hosts. This approach is usually helpful in cases where there are breaks in overall network interconnectivity, for instance if the mobile host becomes disconnected from the fixed network. If the replicated copy of the server is on the mobile host then disconnected operation is possible. With the replicated server objects on the mobile host latency is improved as the use of the wireless link is bypassed. Replication incurs high maintenance overhead in order to

maintain consistency between the object replicas. Replication also undergoes a clean-up stage when multiple replicas of an object are reconciled, or merged back into one primary copy.

Object Migration **-** Replication may not always be the best option for providing high availability and low latency of services for mobile hosts. Another option is to migrate the server object from its original primary host to a secondary host possibly the mobile host. This maintains one single copy of the server object in the system and eliminates the need for any consistency strategy. This mechanism could be used in cases where a pattern arises that shows client accesses have moved from one part of the network to another, migration of the server may help reduce latency. However, it suffers from high setup costs in the same way as replication.

Object Delegation **-** Object replication and migration have high setup costs and may not always be the solution to providing wide availability of server objects. If a mobile host requests the services of a rarely used server object the request may be forwarded or delegated by using a directory service to the host that has the requested server object. Delegation basically introduces a proxy object, the client contacts this as if it was the object it required, but the implementation of the requested object resides elsewhere on another host. This mechanism can be used in the scenario described in the figure above where the mobile host has only to remain in contact with the mobile support station. It acts as a proxy for the server objects the mobile host needs to contact. This mechanism does introduce another level of indirection by having a request pass through a proxy object but it is a simple way of ensuring availability.

Disconnected Operation **-** It is possible to use migration or replication to move server objects to the client. By migration the actual object would reside at the client side and invocations would be local to the client minimising invocations across the network. The objects could be 'pre-fetched' stored in a client side cache. This would require some knowledge of what the user will do what objects it will need to operate if it was to be disconnected from the network. With replication the objects stored on the client side are copies of the original that remain on the server side, but the goal of disconnected operation is the same, the client cache tries to emulate the behaviour of the server.

The following sections review some research projects that used distributed object technology in the mobile computing domain with the goal of improving availability of service to the user. How these projects have dealt with the various issues provides valuable information when addressing the issue of how CORBA can be used in this environment.

2.1.1 Bay Area Research Wireless Access Network - BARWAN

Bay Area Research Access Network is an ongoing project at Berkley that is developing a toolkit for application development specifically for a mobile environment [Katz ‘96]. The motivation behind the project is that mobile applications will require the same remote computing power now available from the desktop. In the BARWAN architecture a network management layer manages connections of mobile units. The applications level has an interface to this management layer, which means that its communications needs are known at all times and it can adapt to the changing network conditions, for instance is informed when the bandwidth has been reduced and can act on this information.

Each mobile device has a proxy, which is a process running on the fixed network that basically manages the wireless connection. It decides what level of encryption and compression is used and performs computing on behalf of the mobile client both interactively and in the background. There are a number of strategies for dealing with the unreliability of the wireless link. One of these supports situations when the mobile host is disconnected. The network management layer will notify the application layer of the disconnection. The application layer can store results until the user reconnects and proxy can forward any changes that have occurred. This is useful if the connection is expensive or unreliable since the user can attach to their proxy and reconnect later to get the results of its computation.

Application and data specific compression also takes place before transmission over the wireless link, which increases the effective bandwidth. Cost models have been developed that can accurately predict the overall latency of each transmission for a given type of data being transferred. For example, depending on the link quality a raw bitmap, a compressed version, or a lossy/highly compressed version of the bitmap could be sent to the mobile client.

They also introduce the concept of 'pre-fetching' data from the server side and storing it in a cache on the mobile host. This can take place when the connection bandwidth is high. When the application is operating in the future it may already have some of the data required stored, this would reduce the demand on the bandwidth for that particular operation.

The project intended to deliver a wireless network providing high connectivity to the mobile applications, through monitoring the performance of the wireless link. When the link performance was affecting the application, some mechanisms to change how the data was sent were imposed.

2.1.2 Bayou - Xerox Parc

Bayou is a project in Xerox Parc aiming to create a platform where applications for collaborative work could be developed [Peterson ‘97]. The type of applications included are shared calendars, document databases, collaborative programming tools and many others. The major goal is to develop the infrastructure to support applications that are specifically designed to enable people to work away from their offices.

Bayou in particular addresses the access to storage systems required by mobile applications. These databases are often shared for both reading and writing. They must be readable and updateable by even those users who may be disconnected from other users. To do this replication is required and Bayou has studied new replicated data schemes and data management issues specific to applications in a mobile environment. Existing replicated data algorithms such as those based on maintaining strong data consistency by atomically updating a set of copies or based on server initiated callback for checking the client cache do not work well in a frequently partitioned network. For partitioned networks the algorithms are usually pessimistic, with locking on replicas, or else provide few consistency guarantees and little support for resolving conflicts.

Bayou went about devising new schemes by first looking at data management issues and application requirements in the context of mobile computing and seeing where the existing schemes fell short. They also decided to build a storage platform, which some prototype applications would use. The following diagram illustrates their architecture:

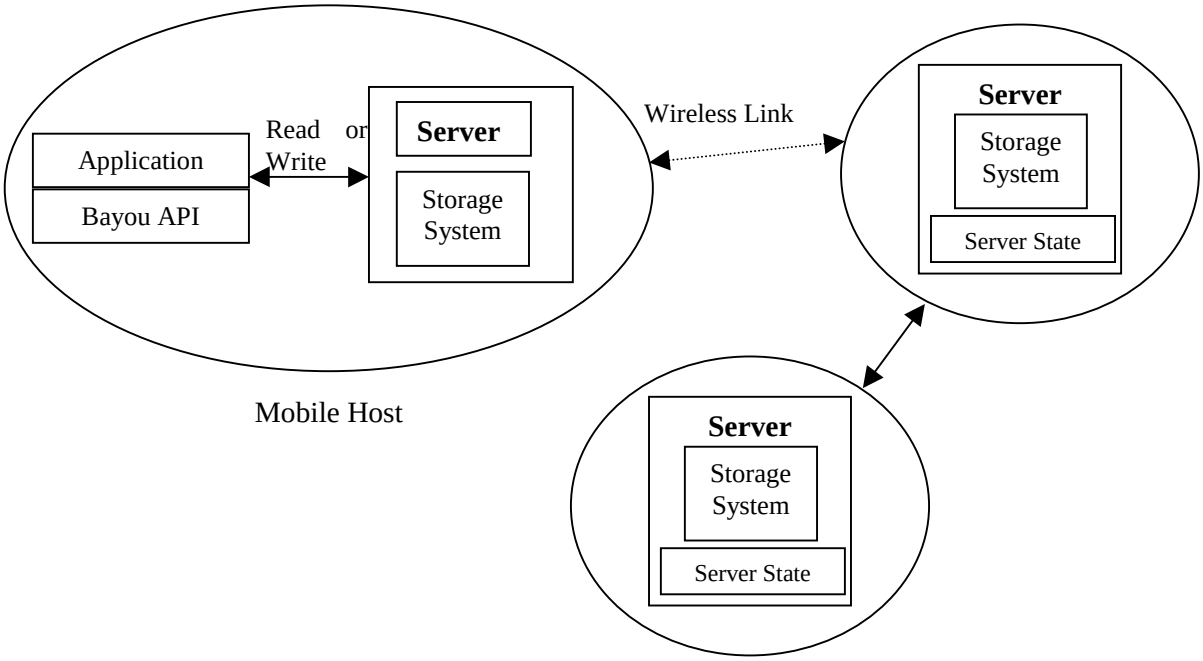

Figure 2-2 The Bayou Architecture

A server may be resident on the mobile host enabling disconnected operation. The server will have been preinstalled if this was the case. Bayou applications can read from and write to any replica without having to co-ordinate this with other replicas. Every replica eventually receives updates from all other replicas through a chain of pair-wise exchange of data. This mechanism allows applications to supply their own data-integrity constraints, conflict detection and resolution procedures and data propagation policies. Bayou servers manage the replication. Each server holds a complete replica of the data. Bayou guarantees that the distributed storage system will eventually move towards a consistent view by having a global order on write operations. The writes carry certain information so that the Bayou server can apply the ones it has received in the right order, without having to co-ordinate with other servers. Reconnection of the disconnected mobile host will have to occur at some stage for there to be complete consistency.

One prototype Bayou application was a group calendar. There are application specific policies that ensure conflicts can be resolved. For example people provide alternative times when

booking a room, this means users don’t have to wait to co-ordinate with other users and seek approval, it also means the users can operate in a disconnected mode.

Bayou presents a system for collaborative applications to work in an environment where users may be disconnected and where finely grained co-ordination between users is not required. This type of application environment matches closely to some of the proposed applications for future mobile computing systems.

2.1.3 Monarch

Applications assume that the characteristics of the network environment remain invariant while the software is in use, however, this isn't the case in a mobile environment. Mobility causes changes in the environment that may result in loss of functionality as well as changes that could result in gains in resources. In Monarch, an OGI project, Physical Media Independence (PMI) was defined, which is an architecture which addresses the management of using the same mobile host application on different networks [Inouye '97]. It answers the question of how a mobile host can be reconfigured transparently as it migrates across different environments and uses heterogeneous network interfaces. The network configuration adapts itself as active interfaces become disabled and new interfaces become available. Adaptation was done intelligently so that each layer of the network informed the adjacent layer what was going on, for instance the transport layer would send a notification to the application layer to inform it that something is changing. Applications care about bandwidth, connectivity and cost so they want to receive notifications about these characteristics.

The Monarch approach looks at how applications can change their behaviour in a mobile environment by knowing what’s happening at lower levels in the network. One area of experimentation looks at how the applications are affected by changing from an Ethernet connection to a wireless LAN connection. The layers of the protocol stack have an adaptation module associated with them that takes care of what needs to be changed to adjust to the new link layer connection. These multiple adaptation layers pass information between themselves so that the application can still function.

Monarch concludes that applications will perform more efficiently if they co-operate with the operating systems, which informs them of changes in the environment. A useful lesson coming from this project is that it is important for applications to be informed of what is happening at the transport level and to let the application decide how it should proceed using whatever pre-defined policies it has.

2.1.4 MosquitoNet

MosquitoNet is a Stanford University mobility interest group that has a number of projects in the area. One area of research has defined the Mobile People Architecture, the MPA [Appenzeller '99]. This places a person at the end point of the communication session rather than the device the person uses. For example, an email message should be directed to wherever the user is. If they are travelling and have their mobile device switched on then it should reach that device. If they are in their office it should go to their office desktop computer. A new layer called the people layer is proposed that will sit on top of the application layer. This layer needs to name people, map people's names to application specific addresses, and route comminations between people. The MPA introduces the concept of routing between people. A 'Personal Proxy', has been defined which can act as a tracking agent or as a dispatcher. As a tracking agent it contains a list of devices the person is currently available at and as a dispatcher the proxy can format whatever information is being delivered into something accessible at the particular device.

This is a very interesting development in mobile computing. While the work in this project may not directly feed into the work in this thesis it does, however, fit nicely into the vision of an open application environment, where the user sees just one application but this can run on any platform, the desktop or the mobile device. This type of open application environment, or as the MPA calls it, the mobile aware application, can definitely be facilitated using CORBA.

Another interesting project in MosquitoNet is NetTimer [Lai '99], which looks at ways of dynamically measuring bandwidth. This is useful in a mobile network since applications need to be able to adapt to changing network conditions, including changing bandwidth. The NetTimer tool uses a number of bandwidth measuring algorithms to give an accurate reading. This enables the application to adjust accordingly. This is another confirmation that the application needs to have some indication of what's happening at the transport level in order to take action to continue to execute or show an exception to the user.

2.1.5 Rover

The Rover toolkit uses a client/server distributed object model that isolates mobile applications from the limitations of mobile communications systems [Joseph '97]. Current work in this project relates to improving the reliability of the systems operation by improving the failure model [JK '96], which addressed client or communication failures and guaranteed reliable message delivery from clients to servers.

Client applications can run on both mobile hosts and stationary hosts. Server applications are assumed to run on stationary hosts and hold the long-term state of the system. To allow applications to run in the mobile environment, which is characterised by limited communications bandwidth and varying computational resources, Rover introduced two concepts:

- Relocatable dynamic objects
- Queued remote procedure call

A relocatable dynamic object (RDO) is an object that can be loaded dynamically into a client computer from a server computer to reduce client/server communication requirements. An example of an RDO could be a simple calendar item with its associated operations or a complex module such as the graphical user interface for a calendar application. Clients can use RDOs so they no longer make invocations on the server objects. This is useful for disconnected operation.

Queued remote procedure call is a communication system that allows applications to continue to make non-blocking procedure calls even when the host is disconnected. The mobile host has an access manager which keeps track of the QRPCs and drains them whenever the mobile host is connected.

Rover has support for reliable applications built in at the client level dealing with software and hardware failures and failures of the communications link while sending QRPCs. It also supports failures at the server side, through logging of QRPCs.

The following diagram shows where these mechanisms fit into the client/server model:

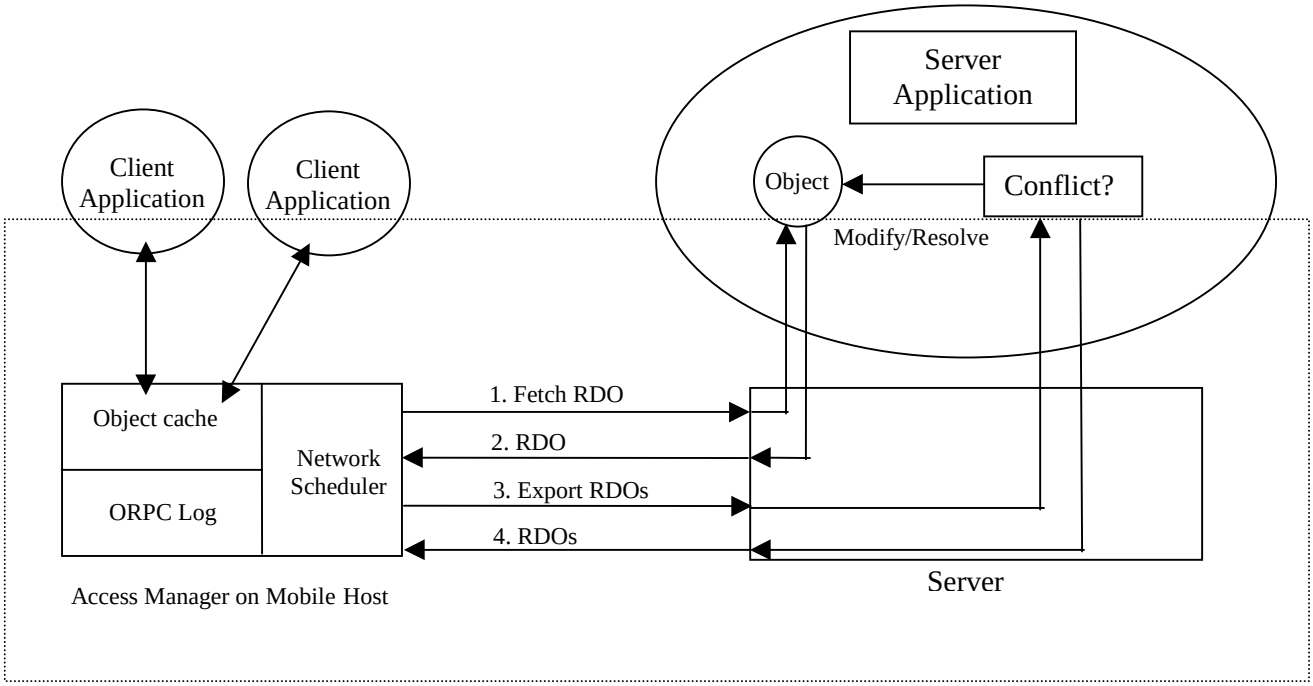

Figure 2-3 The Rover Toolkit client/server distributed object model

The first step is requesting the loading RDOs into the local client cache, this done during a period of network connectivity. The application decides what objects should be pre-fetched and stored and these are passed to the client from the server. The server is responsible for maintaining a consistent view and performs the replication conflict detection procedures. RDOs are passed back to the server according to a predefined schedule. They pass through the consistency checking mechanism and are then returned to the client cache.

This approach of moving the objects from the server to the client to enable disconnected operation formed the basis of the approach of this project. The major difference was that CORBA objects are the units being transfered in this project.

2.1.6 MOWGLI

MOWGLI was a University of Helsinki project with the goal of studying, designing and testing a data communication architecture for a pan European GSM-based mobile data service [Kojo ‘95]. A prototype was developed based on that architecture. Mobility aware applications were developed and deployed as part of the prototyping experiment. The applications were able to operate in disconnected or weakly connected mode and thus mask the inherent problems of using a wireless connection.

The key concept of the MOWGLI architecture was to introduce a new element to the standard client-server paradigm, called the mediator. The client and server communicated with each other through the mediator. The mediator split the end to end connection into two separate parts, one over the fixed link and the other over the wireless link. Each part of the connection could be tailored to its own underlying transport. The following diagram illustrates the Mowgli communication architecture;

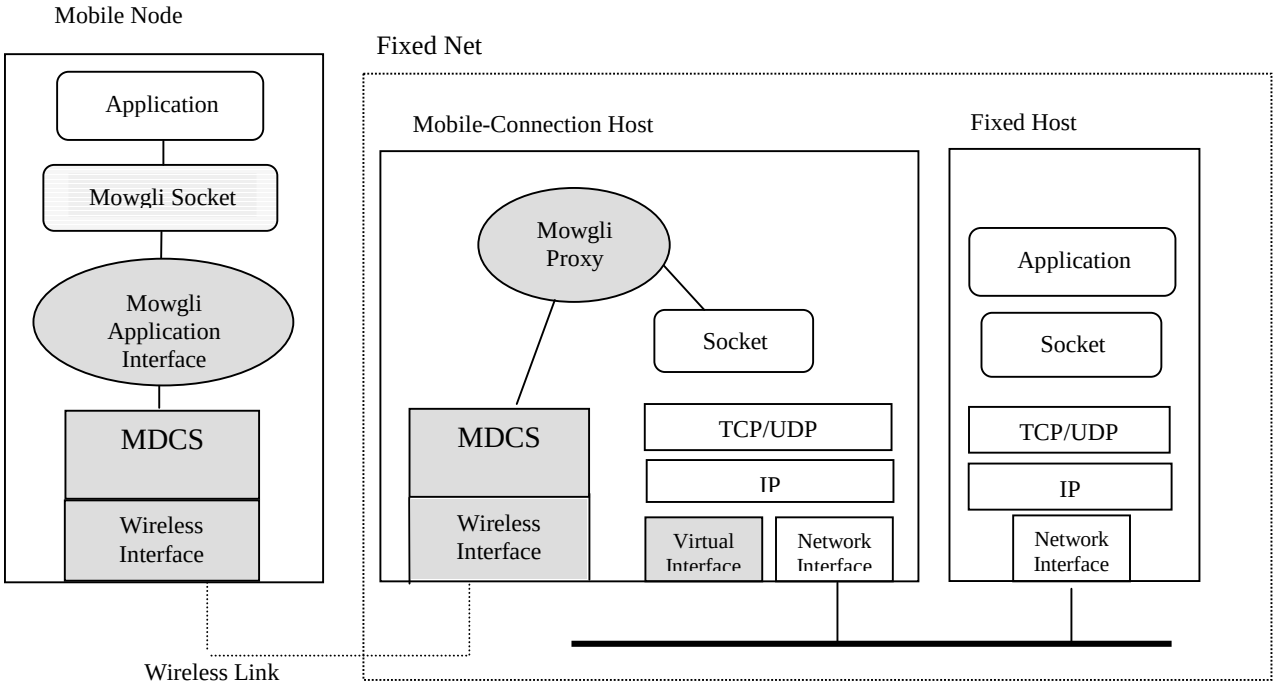

Figure 2-4 The Mowgli communication architecture

The Mowgli Data Channel Service (MDCS) is a special transport service that transparently replaces standard TCP/IP for use over the wireless link. It has a number of mechanisms that improve efficiency over the wireless link such as priority based scheduling of data channels over the link. The Mobile-Connection Host (MCH) in the fixed network provides the mobile host access to the fixed network services. The proxy at the MCH acts as the mediator for all data delivered between an application on the mobile node and the fixed network server. There is also a layer on the mobile host side that sits on top of the transport layer that acts as an interface for applications. MOWGLI also defined a Socket Protocol for communications between the agent on the mobile host side and the proxy on the fixed host side. The Mowgli architecture provides a very clear approach to communications in a mobile environment. The mobile host can communicate with an entity on the fixed network that basically acts as a proxy for the mobile host and translates its requests into something that can be passed onto other

nodes within the network. Between these two entities various strategies can be enacted in order to maximise use of the wireless link capacity. This approach has been used by many projects in this domain and in fact the OMG recommend an architecture like this in their White Paper on Wireless Access and Terminal Mobility, which will be discussed in section 2.3.

2.1.7 Project Review Conclusions

The projects described in the previous sections are just a small number of the total projects investigating issues relating to data communications in a wireless mobile context. It is obvious that the compelling business case pointing to the huge growth of mobile communications and computing in the future has motivated a lot of research in the area. Distributed computing appears to have the necessary tools to deal with many of the issues presented by mobile communications, as illustrated by the numerous projects described above. One approach, object replication, appeared in a number of projects in order to support disconnected operation. It is this approach to support CORBA applications in a mobile domain that is the primary focus of this project. However, before addressing this issue the next section looks at the issues relating to using CORBA in a mobile environment with particular reference to the OMG white paper on wireless access and terminal mobility and their RFP of the same title.

2.2 Wireless CORBA

In the previous section it was shown how many research projects in the mobility domain were using distributed object technology to provide mechanisms and frameworks that would enable applications to operate reducing the affects of reduced bandwidth and increased latency cause by the use of wireless communications. CORBA is an open, platform-neutral technology for building distributed applications in an object-oriented way that hides most of the complexity introduced by distribution. However, CORBA was designed to aid application distribution in fixed networks, it did not have networks that contain wireless links in mind. With the growing number of mobile devices using wireless communications and the diversity of operating systems that will arise from this growth, something like CORBA is required to aid with application development, especially since much of the existing development is using distributed object oriented approaches.

The OMG believes that this is a very important area for CORBA standardisation. In June 1998 the telecom domain task force sent out a request for information for supporting wireless access and mobility in CORBA. Resulting from this the OMG published a white paper on Wireless

Access and Terminal Mobility [OTDTF '98] towards the end of 1998 which outlined many of the issues related to using CORBA in a wireless environment. In May 1999 a Request for Proposals (RFP) was issued that solicited proposals for technology that would allow mobile terminals "to exploit and to provide CORBA-based services" [OTDTF '99]. In the RFP the term "mobility domain" was used to denote a domain that allows access from mobile terminals and provides them with the ability to use and to offer CORBA-based services. This section looks at the issues facing CORBA and the changes that need to be made for it to be used in the wireless domain. The main areas where CORBA needed to address were in relation to providing mechanisms dealing with the unreliability of wireless links and mobile terminal mobility.

CORBA defines standards at three levels, the architectural level, the protocol level and the application level. Distributed objects and their interactions are defined at the architectural level, this will not be affected by wireless network behaviour. How objects from different ORBs developed independently from each other communicate is defined by a set of inter-ORB protocols. These are affected by wireless networks. A set of standard APIs defined at the application level will also require some changes.

2.2.1 Protocol-level Issues

General Inter ORB Protocol (GIOP) defines the minimum protocol necessary to ransfer invocations between ORBs. The Internet Inter ORB protocol (IIOP) is the inter-ORB protocol that runs over TCP/IP and is probably the lingua franca of CORBA, since it gives the widest possible interoperability between CORBA ORBs. For wireless access using IIOP presents problems because of the fact it uses TCP/IP.

Firstly, there is a problem with latencies. In wireless networks latencies are higher than in fixed networks. The throughput of a link is unpredictable and more error prone, this type of behaviour does not suit TCP/IP connections and they perform badly. With the higher latencies a TCP/IP end-point may believe that data was not received at the other end and may retransmit needlessly wasting bandwidth. TCP/IP also assumes that any errors are related to network congestion, this would cause it to reduce transmission rates.

IIOP assumes that connections between objects are maintained continuously. This assumption cannot be made when dealing with wireless connections. Since they are more fragile, the fact that a connection may be broken and re-established is a reality of wireless networks particularly when a mobile device is roaming. It may also be costly to keep a connection

established and the user may want to disconnect but reconnect to the same session at a later stage. IIOP does not accommodate that type of behaviour. To overcome this problem a strategy of wrapping the communication in a way that it would allow disconnection and reconnection would be ideal. This would require a change in the client/server model used by IIOP where the server will cancel any operations it was about to execute on the client if the connection has been lost. The ALICE framework, which is presented in section 2.3 provides a strategy for dealing with this problem.

2.2.2 Application-level Issues

It is accepted that on a wireless network, disconnection and reconnection may occur quite frequently and latencies and bandwidth will fluctuate. CORBA applications need to be aware of these problems and should have more application level facilities to handle this.

One approach to overcome these problems is to have applications that could intelligently handle circumstances such as lost connections or reduced throughput. The applications could deal internally with network conditions, to determine network conditions and perhaps modify how the application itself executes depending on those network conditions. In this way applications would become network aware. Another approach suggested in the white paper is to have a set of policies enforced by the ORB for applications that aren't 'network aware'. These could be used in situations when there is a disconnection or when bandwidth falls to a very low level, which may affect the application executing.

The primary focus of this thesis is to allow continued application operation through support for disconnected operation, similar to the approach used by many of the research projects discussed in the previous section. Further discussion on disconnected operation is presented in section 2.4.

2.2.3 General Issues

The physical movement of the mobile device also introduces a set of problems for CORBA applications. CORBA object references use the physical location of the object as one of the attributes that goes into making up the reference. This reference stays the same for the lifetime of the object, since CORBA assumes the object remains in the same location. However, if the object's location changes it will no longer be connectable. Some mechanism is required which ensures that the object can be found even if it's location changes. This problem will become more apparent as more diverse applications are being developed for the mobile so that the

mobile device will also become host to server applications. There needs to be some way in which clients can get a reference to server objects that will not be affected by a change in the server's location. The ALICE framework defines a way of ensuring clients can remain connected to a server that changes its location, this is presented in section 2.3.

2.2.4 The OMG Wireless Access Reference Model

The OMG has come up with its own reference model in relation to wireless access and terminal mobility. The key entity in enabling mobility is the Access Gateway, which is the gateway between the fixed network and the wireless network. It is this gateway that the mobile terminal contacts. The model assumes that the client and server are in different ORB domains and that the communications path between them includes a wireless segment. The following diagram illustrates the OMG reference networking model;

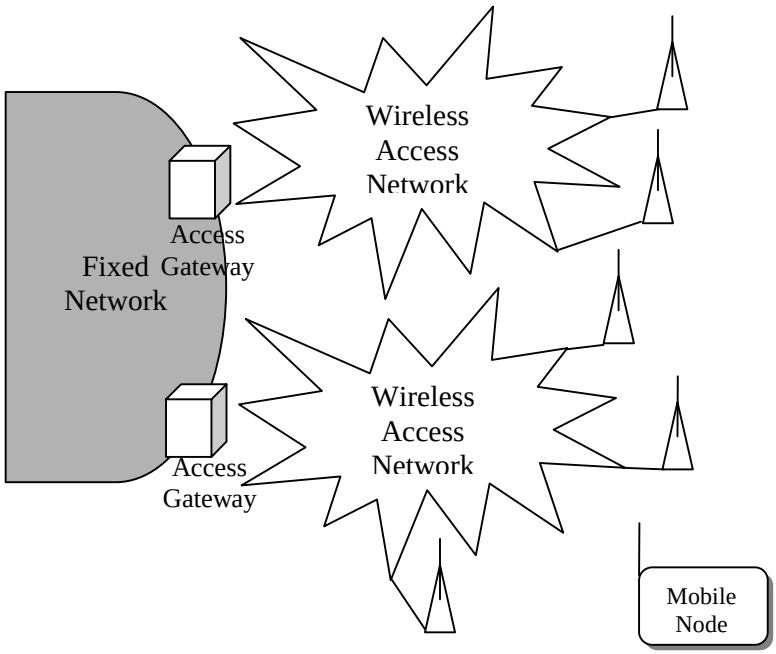

Figure 2-5 Reference Networking Model

The following diagram shows the protocol stack recommended by the OMG, showing how it expects the various nodes in the reference model to communicate over TCP/IP or wireless transport protocols. In the same way that IIOP was developed as the mapping for

communication using TCP/IP it is possible that a GIOP mapping could be defined for the wireless transport and that could be used over the wireless link.

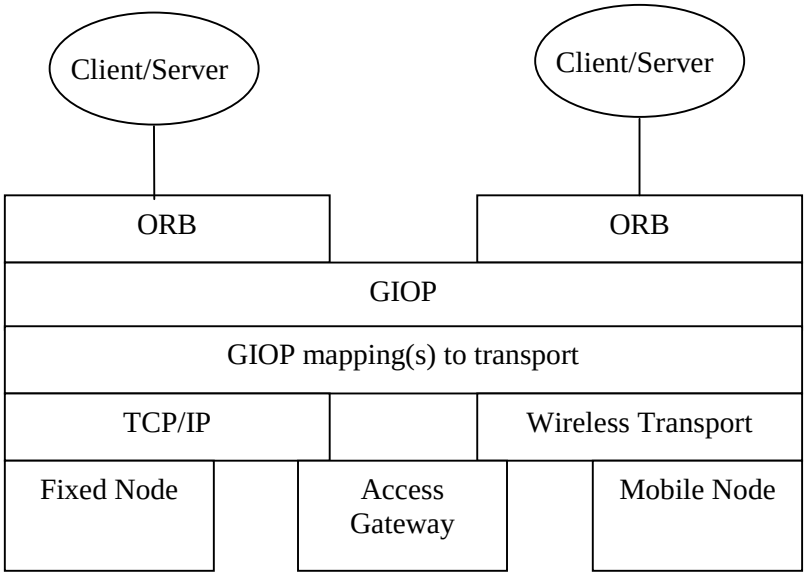

Figure 2-6 Protocol Stacks in the Reference Networking Environment

WAP is an example of one of those protocols gaining in support in wireless networks that could have a GIOP mapping. This would mean that GIOP messages would be mapped to WAP as it is the protocol used over the wireless link. This could possibly result in a mobile wireless inter-ORB protocol being specified for communication across the wireless link, which could be translated into IIOP in the fixed network.

One requirement on the wireless transport is that it can handle temporary disconnections and automatic access recovery. This is something the ALICE framework can do, which will be discussed in section 2.3

A number of projects have been investigating using the OMG concept of the mobility domain, in particular the ACTS project DOLMEN.

2.2.5 Dolmen

Dolmen [Raatikainen ‘97] was an ACTS project that defined and validated an Open Service Architecture for fixed and mobile environments, known as OSAM. Dolmen built the architecture using the Telecommunication Information Networking Architecture Consortium (TINA-C http://www.tinac.com/) Distributed Processing Environment (DPE). It viewed the telecommunications infrastructure as a large scale, distributed processing environment.

Dolmen examined CORBA-Based object communication in the context or wireless access and terminal mobility.

Dolmen looked at two aspects of mobility,

- Personal Mobility
- Terminal Mobility

For personal mobility the TINA concept of the User Agent (UA) was used. It normally gives a user representation in the service provider's domain. As the user changes location Dolmen introduces the concept of a home and a visited domain. So now there are two User Agents each containing information about the user and providing the user with access to the platform.

For terminal mobility the characteristics of wireless communication must also be considered, such as low bandwidth and unreliable communications link. Since the OASM included both fixed and mobile domains one of the goals was to enable transparent computational object communication across the mobile link. The mechanism used to allow this was based on IIOP. In Dolmen an ORB was pictured at either end of a mobile link.

To get the mobile and fixed domain communicating the concept of 'interoperability bridges' was used. Interoperability bridges are described in the CORBA 2.0 architecture. A half bridge existed in each domain that had the functionality to deconstruct and reconstruct communications across the mobile link. The following diagram illusrates the components of the Dolmen architecture:

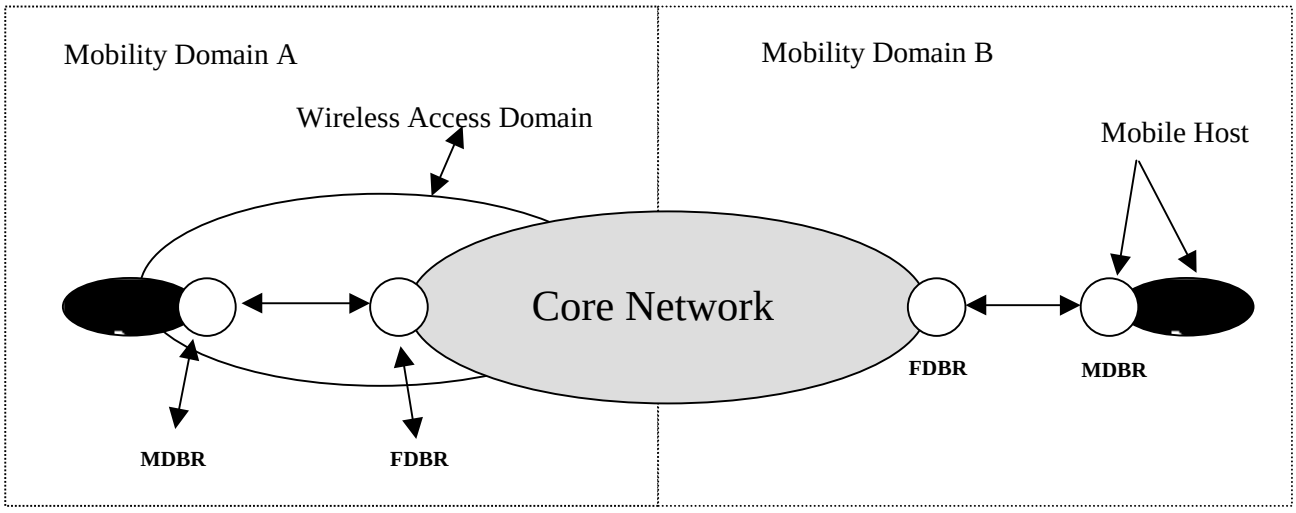

Figure 2-7 The Dolmen Architecture

A Fixed DPE Bridge (FDBR) serves as a DPE access point for mobile terminals. A 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. The MDBR and the FDBR perform location management functions and DPE handovers, enabling terminal mobility on the DPE level.

The project defines a special Light-Weight Inter-ORB Protocol (LW-IOP). This protocol takes the problems of a wireless link into consideration and has efficient message formats and a compressed data representation for object communication. This approach does have a drawback in that the application needs to be able to communicate with LW-IOP. A gateway to IIOP can be defined to allow it to communicate with applications using IIOP. This is an important choice whether to use the lingua franca of CORBA, IIOP, which has such wide usage or use a newly defined IOP. As is described in section 2.4 the ALICE framework still uses IIOP for communication across the wireless link.

DOLMEN implemented two services to run on this architecture, a Hypermedia Information Browsing service, and an Audio Conferencing service. The end user services were able to make full use of the facilities offered by the service machine on the fixed network.

DOLMEN used CORBA concepts such as bridging to deal with the introduction of a wireless link in the overall communications architecture. The project also made the leap of defining a new IOP for use over the wireless link. This choice places demands on the fixed part of the network in that it now requires a gateway to translate this protocol into IIOP. The next section describes the ALICE framework and how it deals with the issues presented by having wireless links in an environment that uses CORBA.

2.3 The ALICE Framework

The ALICE (Architecture for Location Independent CORBA Environments) platform is used to deal with the transport issues relating to using CORBA in the mobile environment. It is specific to IIOP but it could be used for a variety of protocols. It allows CORBA objects running on mobile devices to interact transparently with objects defined within other ORB implementations. ALICE allows server as well as client objects to reside on mobile hosts and provides a mechanism to support the movement of the servers without the requirement of a centralised location register [Haahr '99].

The components that make up the communications architecture in ALICE are shown in the following diagram:

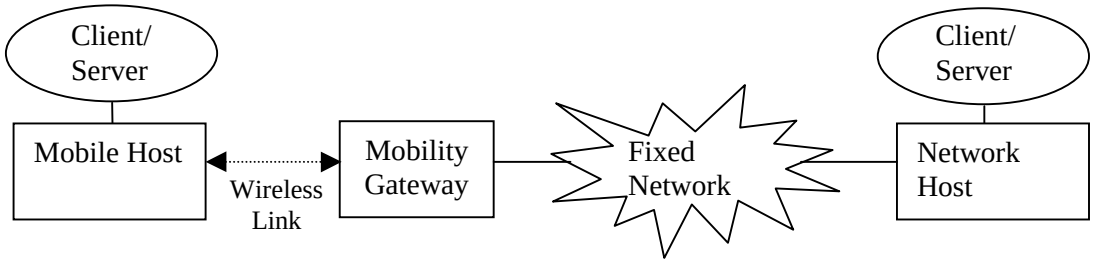

Figure 2-8 Client/Server Communication in a Wireless Network

The Mobility Gateway (MG) acts as a bridge between the wireless network and the fixed network. It takes on a similar role to the access gateway defined by the OMG for the mobility domain. It has a number of roles in the ALICE framework, it acts as a proxy for a mobile host by relaying incoming and outgoing messages over the fixed network for the mobile host. It also performs address translation and redirection for a server on a mobile device that changes location. The mobile host can change location causing it to change the mobility gateway it uses as access to the fixed network. This is a *handoff*. This involves transferring state information from the old mobility gateway to the new mobility gateway and is a difficult process.

ALICE has defined a protocol stack that hides some of the problems related to the use of a wireless link such as low bandwidth and unreliable connectivity. The protocol takes a layered approach in order to provide IIOP functionality as illustrated by the following diagram.

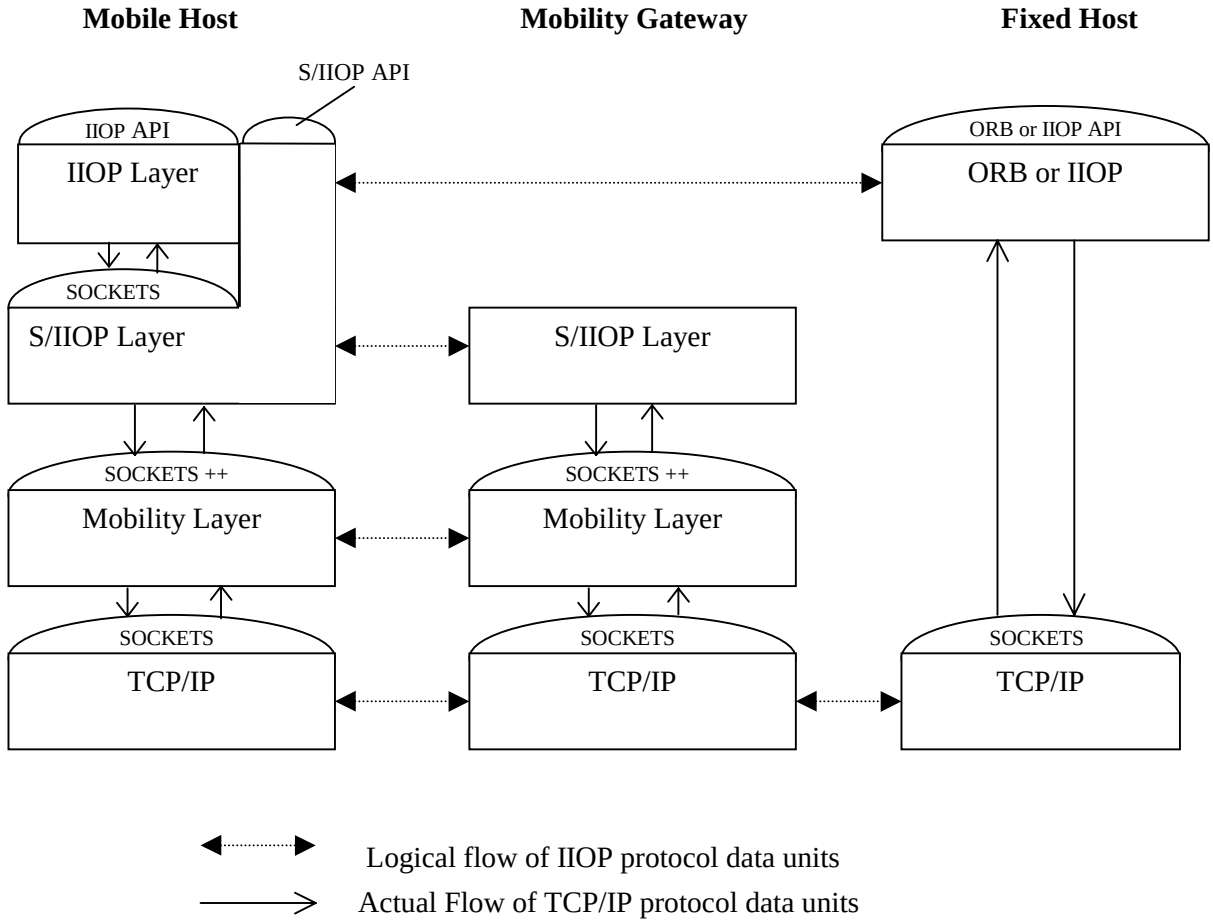

Figure 2-9 ALICE Layered Architecture

2.3.1 The IIOP Layer

The IIOP layer provides the ALICE implementation IIOP. It enables the creation of the eight message types that are required to transparently locate and invoke methods of a server object. For example a method is invoked by using an IIOP *Request* message. There is also an API for the IIOP implementation that was initially defined in OMG's Interface Definition Language (IDL) and subsequently implemented in C++.

2.3.2 The Mobility Layer

The Mobility Layer hides broken TCP/IP connections from the layers above it by providing a logical connection abstraction. As far as the IIOP layer is concerned the connection below it is a normal TCP connection. When the TCP/IP connection is lost the mobility layer continues to

make attempts to reconnect. The IIOP logical connection remain open for this period of time and any request data is cached until such time as the TCP/IP connection is re-established. There is an assumption here that a reconnection will occur within a reasonable period of time.

The mobility layer also allows the IIOP layer to allocate TCP/IP ports on the mobility gateway for incoming connection attempts. This ensures that clients on the fixed network can set up TCP connections to the mobile device by contacting the mobility gateway, which then creates the logical connection with the mobile device. The mobility layer also deals with *handoffs* between mobility gateways when the mobile host moves from one gateway to another. This involves tunnelling open transport connections between fixed hosts and the old mobility gateway for the remainder of their lifetime. The mobility layer provides mobility information for the S/IIOP layer on both the mobile host and the mobility gateway that enables address translation and request forwarding to be performed.

2.3.3 The S/IIOP Layer

The S/IIOP layer is the mobile-aware component of the IIOP implementation in ALICE. It is used together with the IIOP layer to support server objects on the mobile host. It is used by the application to perform operations relating to IORs. It uses the callback mechanism of the Mobility Layer to keep track of the current mobility gateway and uses this information to manipulate the IORs to keep them up to date. This process is known as 'swizzling'. An IOR contains a number of profiles, each specifying a location, in the form of hostname and port number, where the object can be reached. A CORBA server creating an IOR adds a profile to it indicating for each endpoint at which the object can be reached. The client on connecting to the server tries each profile of the server's IOR until one succeeds. Swizzling an IOR occurs when a new IOR is to be created and the mobile host is connected to the mobility gateway. Each profile that refers to the local mobile host is removed and replaced by one referring to the S/IIOP layer of the mobility gateway the mobile host is connected to. The S/IIOP layer on the mobility gateway listens on a default port, which is known by the S/IIOP layer of the mobile host, thus allowing swizzling to occur on the mobile host. When the server application starts to listen on an IOR, a logical connection is made between the mobile host and the mobility gateway. This means that any connection attempts on the S/IIOP layer of the mobility gateway will be forwarded to the mobile host.

ALICE deals with the major recurring issue relating to using a wireless link of how to handle disconnection and reconnection. This has been achieved in a way that is transparent to the

CORBA application. The mobility layer will continually try to reconnect if the TCP/IP connection has been lost. However, if the connection stays down for an extended period of time then an exception will eventually have to be passed back to the application. It is in situations such as this that a way of allowing the application to continue operation even when the wireless connection has been lost would be useful. The application would be further protected from problems relating to the wireless link. The next section addresses some of the issues involved in using a disconnected operation approach such as how it can be supported using object replication and relocation and how to ensure consistency on replicas.

2.4 Disconnected Operation

The CORBA environment is essentially a connection-oriented programming environment, a client must maintain a connection to a server. In a mobile domain the challenge is to maintain that logical connection between the server and client regardless of what happens at the communication level.

What is required is a mobile aware ORB that has techniques for optimising access to server objects, which will deal with the latency and bandwidth problems. It also needs some techniques for disconnected operation, which may include object replication, object caching on mobile hosts and some method of deferred remote invocation when a connection is down.

Some of these techniques overlap in supporting the various requirements. Object caching and object replication can help with optimised access to objects and also in disconnected operation. The following sections look at various existing projects that have used and experimented with some of the mechanisms mentioned. The aim is to use some of these ideas and present some possible solutions supporting mobile operation for CORBA applications in a wireless network.

Replication is one of approaches that can be taken for disconnected operation in a client/server paradigm that were discussed at the start of section 2.1. Replication introduces problems such as consistency and conflict resolution.

2.4.1.1 Using the Disconnected File System Approach

File caching is one approach that enables disconnected operation for extended periods of time in a file system. By using this approach it must be accepted that it will lead to conflicts, which will affect to validity of certain read and write operations. A user of such a system must be aware that a write operation may fail long after it may have been committed on the local host and also a read operation may return some outdated information without showing any errors.

Coda [Kistler '91] is a diconnectable file system. It provides resilience to server and network failure through the use of server replication and disconnected operation. The name-space is mapped to individual file servers as volumes. Each client has a cache manager in charge of caching the volumes. To achieve availability Coda uses replication, allowing volumes to be read and written at more than one server. Coda uses an optimistic replication strategy, which provides high file availability by allowing reads and writes even when the server and clients are not connected. The strategy can be described as *read-status-from-all*, *read-data-from-one*, *write-to-all* [Saty '89]. With optimistic replication there will be conflicts. In Coda write-write conflicts are detected using version vectors. A version vector summarises the modification history of a file and conflicts are detected by comparing version vectors. If the vectors are the same then there is no conflict, if they are different then a resolution process must take place. This is done by a resolution protocol and if it's not successful then by a manual process.

Disconnected operation in Coda takes place when the replicated servers are unreachable and the client enters what is called emulation state where the cache emulates the server. At reconnection the cache volume is merged with the servers volumes. Conflicts are detected as described above.

Disconnected operation may only be acceptable to the user if reconciliation of conflicts is automatic, and is done in a predictable, repeatable and acceptable way. One wireless experiment has defined a number of algorithms for automatic resolution of conflicts based on priority assignments and invalidation [Hild '95]. It was illustrated that total reconciliation can be performed as a sequence of partial reconciliations.

At the University of Washington a disconnected operation programming environment for mobile computing devices was developed based on experience building a disconnected NFS for portable computers. Concepts from the CODA file system approach were used to develop a proxy to add to the standard client/server scenario. The proxy was used to mask mobility. It acted as a pseudo-server to the client and a pseudo-client to the server. In the connected state the proxy forwarded all of the client requests to the real server. In the disconnected state the proxy emulated the real server but the proxy maintains a log of modifying operations that were to be reflected on the server on reconnection [Fiuczynski '95].

2.4.1.2 AspectIX

AspectIX [Geier '98] is and open architecture defined to overcome CORBA's limitations when it comes to providing an open architecture for use in a mobile environment. It moves beyond the static client-server relationship of CORBA and uses the concept of distributed objects. In this system an object is made up of distinct fragments and each fragment communicates with other fragments to achieve the desired behaviour. The client in this architecture is a little different from the CORBA one where the local object is a stub that delegates invocations to the server object. In AspectIX the client of the object always has one of these fragments in its local address space. The fragment could be a simple stub as in CORBA. The stub may connect to another fragment (like the server object in CORBA) that implements the object's functionality. The fragment at the client side may also be a little more intelligent and may cache some of the object's data or implement some of the object's functionality locally. This kind of behaviour would be very useful in a mobile communications environment.

For replication the distributed object is extended by an additional fragment, which just acts as a replica. The replica has the responsibility of implementing the consistency model communicating with other fragments. For mobility the distributed object is extended with a new fragment at the destination site. The state of the original fragment is then transferred to the new one and the old fragment can be replaced by a simple stub acting as a forwarding entity. The original fragment can then be deleted, this results in the migration of the distributed object. The distributed object makes the decisions about replication and mobility based on the requirements of client.

This mechanism could be a useful way of extending CORBA's static client server architecture and allow it to function in a mobile environment using the tools of replication and migration in a distributed object scenario.

2.4.2 Using Disconnected Operation In CORBA

In supporting disconnected operation in a CORBA environment it is obvious that the mechanisms of replication and relocation will be required. The next section describes how disconnection can be integrated with the ALICE framework and proposes the mechanism of Object by Value which was used as the approach in this project for moving objects around to enable disconnected operation. However, as the previous sections pointed out there is also a penalty of having to provide some support for conflict detection and resolution.

Chapter 3

3. Design

The previous section introduced the concept of how disconnected operation could be supported in a distributed object system. Tools such as object replication and object relocation were used in many of the relevant research projects discussed. In this section a design for integrating support for disconnected operation within the ALICE framework is presented. It describes how the standard CORBA client server model has to be altered in order to support disconnected operation.

3.1 Supporting Disconnected Operation in ALICE

In its present form ALICE always assumes in the case of a transport disconnection, that a reconnection will be possible within a short period of time. For example during a hand-over a connection may be temporarily lost but ALICE keeps trying to restore the connection and when it does communication can continue. The IIOP client and server are never aware that a broken transport connection occurred. ALICE does not consider the case where a transport connection is broken and it is unlikely that the connection will be re-established for a longer period of time. In such a case it would be difficult to hide the communications breakdown from the client and server. If the logical IIOP connection is broken ALICE has no support for resuming the broken connection over a different transport connection.

To extend the ALICE architecture to support disconnected operation it must be decided at what level to introduce this new functionality. For the client side of the application there are two options: knowing that a disconnection in the underlying transport has occurred, or being unaware of the behaviour of the underlying transport. At present the ALICE IIOP clients are 'disonnection-unaware'. To continue this support for cases where disconnections remains for periods of time that would be noticeable to the client a new layer could be introduced to the ALICE framework. This layer could transparently deal with the extended disconnections by providing some additional functionality to allow the client to continue operation normally. There may be cases where it is better for the client to know that there is a problem with the underlying transport causing a prolonged disconnection from the network. In this case the client could be provided with support to continue operation in disconnected mode, but it would be aware of this i.e. it would be a 'disconnection aware' mobile client. For the server side it

would be good to limit the changes to applications needed in order to support disconnected operation. This would mean that existing applications could continue to be used in mobile scenarios with minimal changes. The changes to the server side will be related to whatever decisions the mobile host and client side need to make in order to support disconnected operation.

3.1.1 Using the Rover Approach

One possible scenario for supporting disconnected operation would be to copy the server side objects that a particular client uses from the fixed network to the mobile host. This is the approach taken by a number of the projects described in chapter 2 and in particular the Rover project from MIT where the communication abstraction was changed in order to compensate for the harsh conditions of a mobile environment.

Essentially the client should be able to:

- Store copies of server objects it uses in a cache on the mobile device
- Make calls locally to the replicated objects in that cache

This will insure that client operation is unaffected by network conditions. Obviously there are requirements on this sort of behaviour both for maintaining consistency between replicas and merging copies at strategic points in the operation of the service. There are questions as to whether it is worth the effort in downloading the server side objects just because the network conditions may sometime prohibit normal application behaviour. Moving objects to the client side also raises the questions of whether the mobile host operating system can support the objects and the implementation code in order to continue operating the service in the same way as it was when it resided on the fixed host. A scaled down version of the objects may be required, meaning the objects copied over to the client would contain a subset of the functionality of the fixed version but be sufficient to support operation of the application for the period of disconnection. This introduces the problem of how to determine what that scaled down version of the server objects should provide. The subset of functionality could be client initiated or determined by a profile of the client at some previous time of subscription to the service. In addition there is an important question of how to move the implementation code for those objects to the mobile host and subsequently link the code so that it is executable on the mobile device. In favour of having local invocations will be the improved application behaviour since response times will be quicker. There is also the point that there will be no network connection cost, which may be significant for mobile networks.

Another possible scenario would be to queue invocations while the mobile host was disconnected from the network. This means that invocations now become asynchronous. Operations can continue on the client side even if a response has not arrived for a particular invocation. On reconnection to the network the queued invocations would be flushed to the server side in sequence. This approach means that the mobile host does not need to have any server side application support, which may be significant in terms of the mobile device having restricted relating to its processing power and storage capacity. However, one drawback is that a disconnection may last for an extended period of time and the user will have no feedback on any of the operations that have caused invocations to be queued.

The approach of object replication does raise some issues with the standard CORBA client/server-programming model. Firstly is it possible to make copies of CORBA objects and move those copies from their original location while still being able to set up a communications channel between the client and server in the normal way? If it is possible to move the server side objects how are copies of these objects kept consistent? The next section looks at integrating mechanisms to support this behaviour with the existing ALICE platform and this is followed by some discussion as to how the standard CORBA model can be modified by using some of the tools in the latest OMG specifications.

3.2 Extension of ALICE Protocol Stack

A client in the ALICE environment is shielded from the mobility support that exists in the mobility layer and uses the IIOP layer in the normal way. As far as the client knows it always has a connection to the server, so in effect it is a mobile-unaware client as was stated in the previous section. Initially, the goal was to continue this transparent disconnected operation support, enabled through the introduction of a new layer in the architecture. However, it was decided that it would be better for some clients to know that they were operating in a disconnected mode. Thereby the clients would be aware that any actions they take are tentative and may require future intervention on reconnection as part of a resolution function. To fit in with the terminology already used by ALICE the new layer was called the Dissconnected IIOP Layer (D/IIOP). D/IIOP is an abstract design for allowing disconnected operation in CORBA. In effect the D/IIOP layer intercepts client calls analyses them and begins the process of enabling disconnected operation. The following figure illustrates where the D/IIOP layer fits into the ALICE framework.

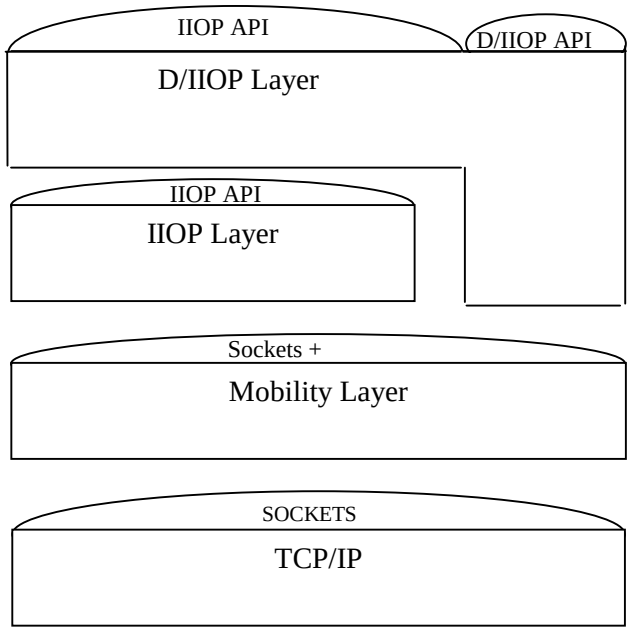

Figure 3-1 Extended ALICE Protocol Stack

The D/IIOP layer exposes the same IIOP API as the IIOP layer. This ensures that the IIOP functionality is still available in the same way to the client application code as it was originally. There is also an additional API, which provides the interface to the disconnected functionality. In this way clients are aware that they are disconnected when they start using this API. This approach was used to allow applications to avail of the existing IIOP implementation but also leave the option to use the new disconnected functionality. The following sections describe in more detail how the disconnection support will work, and certain issues that arise in introducing this functionality.

3.2.1 Replication and Caching Strategy

The basic principle behind what the D/IIOP layer aims to support is based on work originally coming from the file system world such as that of Coda, as well as work from the distributed object technology world such as Rover and Bayou. In Coda availability is achieved through server replication. It was decided to use an optimistic replication approach with the D/IIOP layer. Availability is ensured if updates are allowed on any replica at any time. A user of a mobile device would still want to use their applications independent of any other user in the system. With solutions such as primary-site or majority-vote replication availability becomes restricted [Sorensen’96].

In the same way as volumes or files are moved in the Coda system, server objects need to be moved in the CORBA scenario to a local cache at the client on the mobile host. There are a number of questions that arise in this scenario;

1. When are replicas of the server objects made and passed to the mobile host?

    a) One possibility is to wait until network conditions deteriorate to a stage where a disconnection may occur soon and move the objects at that point. However, this may not be feasible since network conditions have deteriorated, the process of transferring the objects may compound the bad conditions and a transfer may not happen successfully before there is a complete disconnection.

    b) In the Ubidata framework [Afonso ‘98] two different ways of getting replicas to the client were used. The first was to use a periodic pull approach whereby a client (subscriber) uses polling to obtain data from the server (publisher) according to a predefined schedule for each of the items the client is subscribed to use. This can be termed as a pre-fetch approach. The other approach used in Ubidata is an event-driven push delivery of items to the client. The client receives a notification that an item has changed and it can then download that item.

    c) An alternative is to copy the objects when the application is started up on the client side, essentially a pre-fetch approach. The underlying support provided by the D/IIOP layer should identify the server objects required by the client to operate and make copies of these objects. The server could use a client profile to determine what objects it uses. The objects are then passed to mobile device stores them in its cache.

2. How are replicas of CORBA server objects made?

    One of the goals of the project was to find a CORBA compliant solution enabling disconnected operation. It was important to assess the possible ways to create replicas of CORBA objects.

    a) The CORBA LifeCycle Service [LIFE ‘97] defines services and conventions for creating, deleting, copying and moving objects. A client that wishes to move or copy an object issues a move or copy request on an object supporting the LifeCycleObject interface. This puts a requirement on the server side objects, to support that interface. It doesn’t allow objects to be moved like parameters so copying requires the use of a factory service.

b) The project AspectIX discussed in the previous chapter uses a fragmented object representation that is CORBA compliant to enable to creation of copies of CORBA obects. The distributed object is simply extended by an additional fragment that just acts as a replica. It is transparent to the client whether it accesses the remote object or a local replica.

c) Object By Value is part of the CORBA 2.3 specification [OBV ‘98]. It essentially provides the same capability that the 'pass by value' semantics of standard programming languages do. Objects can be moved easily as a parameter in an application.

This is just a short list of the possible approaches to addressing this issue. Further discussion about these options is presented in the server side support section later in this chapter. The choice of which one to use for replication can be put down as an implementation issue.

3. How are the replicas passed to the mobile host?

   This really ties in with what mechanism is used to create the replica. It is important to realise that the object implementation also needs to be present at the mobile host, this places a requirement on the mobile host to support that implementation.

3.2.2 Cache Consistency

Obviously when using replicated objects the issue of consistency between the replicas arises. There are a number of approaches that were considered for ensuring replicas are consistent;

1. Write Through; for any invocation on an object there would be a write through to the original primary copy. Any change on a local copy would be immediately apparent on the original copy on the server side. However, this introduces the possibility of blocking in the system, when one client is writing the changes to the server, other clients will not be able to access the primary copy. It also does not take into account the probability of disconnection, when a write through would not be possible. Optimistic replication was chosen for D/IIOP in order to allow updates on any copy at any time.

2. As with Coda a callback system could be used. The server is responsible for maintaining consistency so it manages when replicas are checked and conflicts resolved. This approach would be suited to the case where clients are mobility unaware.

3. Since in D/IIOP the client is mobility aware it was decided to leave the decision for timing of the consistency management with the client. This could be done automatically at a particular instance, for example when the client reconnects to the network the replica objects it modified could be passed to the server and the conflict detection and resolution process begins.

   The server is responsible for conflict detection and resolution. This will be an application specific process, however, as is the case with Coda and Rover, a system of version vectors should be used to aid the process. A version vector summarises the modification history of the object. If a conflict has been detected the relevant clients should be informed and resolution should take place based on some communication between them.

3.2.3 D/IIOP Protocol

The following diagram illustrates what takes place within the D/IIOP layer in order to enable disconnected operation;

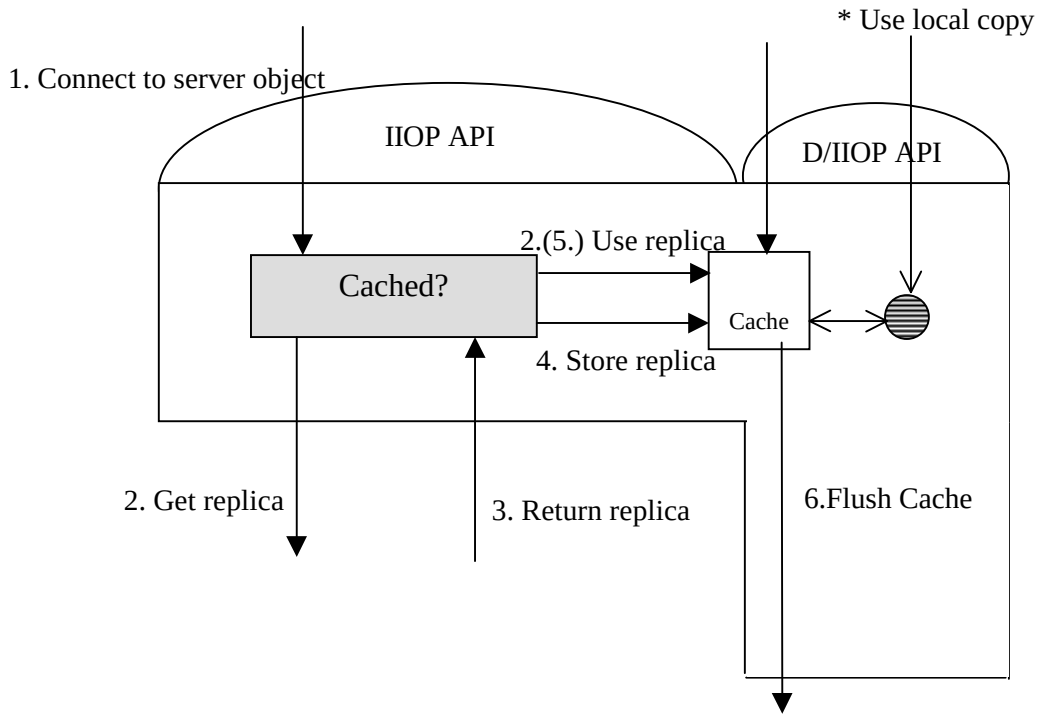

Figure 3-2 D/IIOP Layer

1. Initially when the client applications starts and needs to connect to a remote server object the D/IIOP layer will check to see if the target object has already been cached.

2. If the object was not in the local cache a request is made to the server to connect to the object and make a copy of that object and pass it to the client side.

   If the object was stored in the local cache then the client can immediately start using the stored version.

3. The server side should respond with either an exception to say that the object cannot be copied and the remote version must be used, or it should pass a copy of the object to the client. The remote version may be unavailable due to some security restrictions. It is possible that a response at this stage will indicate that the connection to the server is broken. This information would be provided by the ALICE mobility layer. The client could then be informed to try again later or the client invocations could be queued and passed onto the server when the mobility layer informs the D/IIOP layer that the connection is available once again. When to inform the client that the connection is unavailable should be an implementation choice.

4. The object passed back should be stored in the local cache.

5. The client should then be able to access the stored local copy of the object and continue operation.

6. When the mobile device has been disconnected from the network for a period of time and there are objects in the cache there is obviously a requirement on resolving the cached copies with the remote copies when re-connection takes place. A message will be delivered from the mobility layer indicating that the connection to the remote server side is back up so this should initiate the resolution process. Another option that should be supported is that at a certain point in time determined by the client the cache can be flushed to the server where conflict detection and resolution can take place. This is one function of the D/IIOP API.

The D/IIOP layer needs to provide an Interoperable Object Reference (IOR) translation mechanism if the standard IIOP layer is to be used. The client originally connects using an IOR for an object on a remote device, the new copy of the remote object will have a different IOR if it is copied to the mobile host device. The swizzling layer functionality of the S/IIOP layer could be used.

* The client could directly access the copied object without going through the IIOP layer this raises the question about where the cache is situated. This is an important question in terms of how it relates to the marshalling of calls. In general a CORBA call goes through the

client proxy and then over IIOP to the remote server side. Does this mean that the server side object replicas should be placed below the IIOP level so that the same IIOP calls can be made to them? This would mean marshalling the message so that it can be sent over IIOP and then unmarshalling the message again before it is delivered to the server side object replicas stored in the local cache. A way to avoid this sort of computation on the mobile device would be to have an application level interface which checks to see is the object is cached; if it is then the application can access the object locally without having to go over IIOP. This removes the need for the marshalling and unmarshalling of the original message. This will then remove any transparency to the client, the client will know it is directly accessing a local copy since it is no longer using IIOP.

3.3 Server-side Support

There are two major requirements on the server,

- Make replicas of the server objects requested by the client and pass them to the mobile host where the client resides.

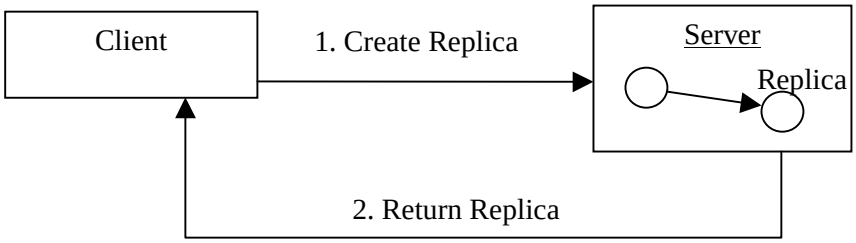

Figure 3-3 Create Object Replicas

- Perform conflict detection and resolution.

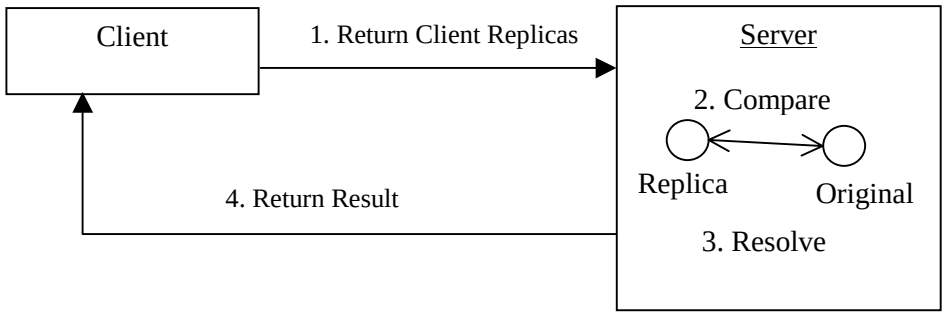

Figure 3-4 Conflict Detection and Resolution

In dealing with requests from the client to make copies of objects the server has to make a decision as to whether it can make a copy of the particular object and pass it over to the client cache. There may be security issues involved. For instance in a banking application it is necessary to know whether a client is the authorised account holder if they request a copy of the account object. When the decision has been made to make a copy the server may need to know some information about what operating system the client has and whether it can use the object it will pass over. It may be necessary to tailor the object to fit the mobile device, this may mean the new object to be passed will have a subset of the primary objects methods and attributes. A possible was of providing for this is by using the Portable Object Adapter (POA) [POA ‘98]. The POA is designed to allow programmers to construct object implementations that are portable between different ORB products. This would suit the case of a mobile environment where different mobile devices using the same servers may use different ORB products. The POA also allows a single servant to support multiple object identities simultaneously and a mechanism for associating policy information with objects implemented in the POA. The servant manager can have a number of implementations for the same object; one of these could be the implementation that will fit the mobile device. When a request is made for a copy the servant implementation for the mobile device could be passed back.

In order to make a copy of the object the Lifecycle Service could be used as was pointed out in an earlier section. It defines services and conventions for creating, deleting, copying and moving objects. The lifecycle service addresses a number of questions in relation to moving or copying an object; what entity does the client communicate with to copy or migrate the object? How does the client find that entity? What happens to the implementation code of a copied or migrated object? A client that wishes to move or copy an object issues a move or copy request on an object supporting the LifeCycleObject interface. This puts a requirement on the server side objects to support that interface. They are referred to as the target objects.

The POA and Lifecycle Service are two possible standard CORBA approaches to supporting the server side functionality. Object by Value could provide the necessary server side functionality. This specification allows the passing of objects by value rather than by reference. It was not developed specifically for complex caching and replication, however, this can be built on what OBV offers. It works in a similar fashion to standard programming languages' pass by value semantics. A new instance of an object is sent from one entity to another, the new object has a separate identity from that of the object on the sending side.

3.4 Summary

This chapter began with a discussion of supporting disconnected operation in ALICE. A proposal of and approach for ALICE similar to that used by Rover was presented. The D/IIOP layer would provide the necessary functionality to get copies of server objects onto the mobile host. A discussion of the possible mechanisms that could be used to implement this abstract design was also presented. The implementation of some of the aspects of this design is presented in chapter 4.

Chapter 4

4. Implementation

This chapter describes the implementation carried out in the project. Firstly, the goals of the implementation are presented. This is followed by a number of sections dealing with various implementation choices, the implementation language, the ORB implementation used, the choice of Object-By-Value (OBV) to implement aspects of the design from chapter 3 and the decision to use the Java Native Interface to enable integration with ALICE. This is followed by a description of the sample application used in order to evaluate the design of chapter 3.

4.1 Implementation Goals

The goal of the implementation was to take the design for disconnected operation presented in chapter 3 and implement as many aspects of it as possible in order to investigate the implications of using this approach. Part of this process was also to integrate the D/IIOP functionality into what already existed in the ALICE framework this would mean that a client could have all the mobility features offered by ALICE with the extra added feature of being able to operate in a completely disconnected mode.

4.2 CORBA and Java for Mobile Objects

The Java programming language is very suited to writing network programs [Java '97]. It is easy for Java applications to send and receive data across the Internet. By using Java and its suitability to networking, in this case the network being a wireless one, the implementation of the disconnected functionality described in chapter 3 would be made easier.

Java also suits the concept of moving objects around. In the Java world mobile objects are defined as objects that move between two or more applications. Both the state and the code of the object move from one application to another. In Java, moving the state of the object is a straightforward procedure because of a facility built into Java called object serialisation [MOJ]. Object serialisation provides a default automatic mechanism for reading and writing the state of an object to a data stream. Java also has a feature called class loaders that enable the actual code to be moved to where the state was moved so that the complete object can be used at the new location. Class loaders are used to locate and load the bytecode for a Java class across the network. This is particularly suitable for the case of a mobile host where it is more

probable that the required classes do not reside on the mobile host. How the class loader was used in this project is described in a later section of this chapter.

Using CORBA with Java provides a powerful tool in terms of mobile objects. Some of the best features of Java can be used with CORBA in the development of CORBA application. One example of this would be CORBA using Java's garbage collection in order to control the cleanup of stubs on the client side. Another would be that some ORBs support is the passing of serialised Java objects between two CORBA applications. This is through the Java mapping of the OBV specification. The facility to move code around can also be used from within CORBA applications. How this specification works is described in the next section. However if a CORBA application was supporting several languages simultaneously it could not support the passing of code for all objects. Supporting mobile code for some languages, such as C++, is very difficult. The receiving application either has to have compile-time knowledge of the code being downloaded, or has to be able to acquire it, maybe in the form of a dynamically loadable library (DLL). Although the design does not require a particular language implementation, Java was chosen for its built in object mobility support.

4.3 Object-By-Value

OBV [OBV '98] was the chosen approach to allow the replication of CORBA objects and therefore enable the support for disconnected operation described in chapter 3. It appeared to be a more direct way than the LifeCycle service through which clients could get copies of server objects passed to them. The LifeCycle service appeared to place more requirements on the server implementation since it had to inherit from the LifeCycle. Pure CORBA (2.1) objects cannot be passed by value, only the passing of object references is supported. CORBA objects are defined by an IDL interface. This allows any type of implementation, which is valuable in a distributed system. However, there are times when it is useful to pass an object by value rather than by reference, and many examples of passing objects has been described in great detail in chapter 2. To support this concept in CORBA where the receiving side receives a "new" instance of the object, with a separate identity from the sending side extensions were made to CORBA and to IDL. The notion of a ***valuetype*** was introduced into CORBA. In addition the notion of an ***abstract interface type*** was also introduced which allows a developer to specify if an operation can explicitly support receiving either a valuetype or an interface at runtime. A primary goal of the new extension was to provide good support for Java users of CORBA. It is felt that a key factor to the continued adoption and deployment of CORBA will be the ease of interoperability between Java and other language platforms. Because of this the

OBV extension was designed so that it would be easily implemented in Java. It was because of the ability to move objects around and because of its close link with Java that OBV was used to implement the functionality to support disconnected operation.

4.3.1 Valuetypes

Valuetypes are the key new component to CORBA that allows the passing of an object by value. They are in many ways equivalent to regular IDL interface types but they are also an indication to the developer that they will have some extra properties; they will have state associated with them and also an implementation that will be required to move. These properties put extra requirements on the valuetype beyond that of a normal IDL type. It is important to remember that valuetypes are not CORBA objects. CORBA objects are not being moved around by using valuetypes [Vinoski '98]. They do not have IORs associated with them. It is also important to note that when using a valuetype the local implementation is always used, there will be no remote invocation. Invocations do not pass through the ORB. This is an important development for the implementation of the D/IIOP layer. The client will not be able to make calls through the IIOP API once the object has been passed to the client side. The calls will all be made directly on the Java object implementations of the CORBA valuetypes.

ValueBase - All valuetypes have a conventional base type called CORBA::ValueBase. This is a type that plays a role similar to CORBA::Object i.e. it supports the common operations available on all valuetypes. In the language mapping the ValueBase is mapped to an appropriate base type that supports the marshalling and unmarshalling protocol.

Supports - The notion of an abstract interface has also been defined and this allows the application at runtime to decide whether to use the actual valuetype or the object. A CORBA object can inherit in the normal way from the abstract interface, but the valuetype will 'support' the interface. This will require the valuetype to provide implementations for any of the methods defined in the interface. The following sample IDL illustrates some of the new IDL features [DOG]:

```
//Define an abstract interface
abstract interface AnAbstractInterface{
     void print();
};
//Define an interface that inherits from the abstract
//interface
interface Example : AnAbstractInterface{};
```

```
//define a valuetype that supports the abstract interface
valuetype valueExample supports AnAbstractInterface{
      // a public state
      public string name_state
};
//define an interface that uses the abstract interface
interface ValueExchange{
      //this operation returns AnAbtractInterface by value
      //or by reference
      AnAbstractInterface getInterface(in boolean byValue);
};
```

In the example above the client will be able to connect to the ValueExchange object and invoke the method getInterface to download either the reference for the Example object or the state of the valueExample valuetype. The valueExample will have an implementation of the print() method from the abstract interface and the client can invoke this method if the value has been passed to it and if the code is local. This will be a local invocation. The Example object will also have an implementation of the print() method. The client can invoke this method if it has the reference to the Example object. This is a remote invocation.

Loading - When the valuetype has been moved the new location expects that it can invoke operations. To do this it requires the code. If it currently holds an implementation class, then this is no problem. If it does not hold an implementation to reconstruct the object it must attempt to load the code of the object. It can do this remotely in Java and other portable languages, it will usually have to be a local process in C/C++. If it cannot do this then it must raise the NO_IMPLEMENT exception.

Creation and Factories - When an instance of a valuetype is received by the ORB, it must be demarshalled and an appropriate factory found for its actual type so that the new instance can be created. The type is encoded by the RepositoryID, which is passed over the wire as part of an invocation. The mapping between the type, which is specified by the RepositoryID, and the factory is language specific. The application must register a value factory with the RepositoryID value before an attempt is made to unmarshall an instance of a valuetype. If the factory can't be found then the ORB raises a MARSHAL exception.

GIOP/IIOP Extension - The general approach in extending GIOP and IIOP ws to add support for the data, in other words the state, and support for the transmission of the type information. The actual transmission of the code is outside the scope of the IIOP definition but it carries enough information to support it.

Java Language Mapping - The mapped valuetype must implement the standard Java interface java.io.Serializable. The mapped Java class contains method definitions, which correspond to the operations defined on the valuetype in IDL. These definitions are defined by the developer of the class in Java. As was stated before the actual code for the methods must be provided before the mapped valuetype can be used.

Initializers - This provides a hook for the application to construct an instance of the valuetype. They are equivalent to constructors in both C++ and Java.

4.3.2 JavaORB

As OBV is a relatively new extension to the CORBA specification there are not many ORB implementations available that fully support it. The TAO ORB from the University of Washington (http:// partially implements the specification and has plans for comprehensive support. ORBacus from Object Oriented Concepts (http://supports it in their alpha release version 4.02, however, this is a C++ implementation. Thus it was decided to use the implementation from the Distributed Object Group (DOG), called JavaORB (version 2.0) since it has a near complete Java Implementation of the specification, which is also free [DOG ‘99].

4.4 Integration with ALICE

One of the goals of the project was to integrate the new disconnected functionality with the existing ALICE framework. The ALICE framework included an IIOP layer [Cunningham '98], which provides the software components that allows a developer to build applications that can communicate using IIOP. The IIOP layer implemented the IIOP protocol through a set of easy to use objects to send and receive IIOP messages and create IORs. The IIOP layer was implemented using C++, which meant the API was C++. Since the D/IIOP layer was to be implemented in Java a decision to use the Java Native Interface [JNI ‘98] to access the functionality of the IIOP layer was made. The JNI allows the developer to invoke methods in another language, in this case C++, from a method within the Java code called a native method, which is identified using the 'native' keyword. When this keyword is used no body for the method is declared within the Java class in which the method is declared. Instead, the body of the 'native' method is contained in a separate C++ source file. The benefits of Java for mobility discussed in a previous section do not outweigh the overwhelming task of re-implementing currently tested and debugged C++ code.

The D/IIOP layer IIOP API would now act as a proxy for the IIOP layer API. The objects in the IIOP layer API would have an equivalent in the Java version of the API. This would require some mechanism for accessing the constructor of a C++ class from a native interface call and returning a reference to the corresponding Java object implementation so that the application code can continue to invoke methods on the object it created. The following diagram shows the Unified Modelling Language (UML) class diagram for the objects in the IIOP layer for GIOP message representation and marshalling.

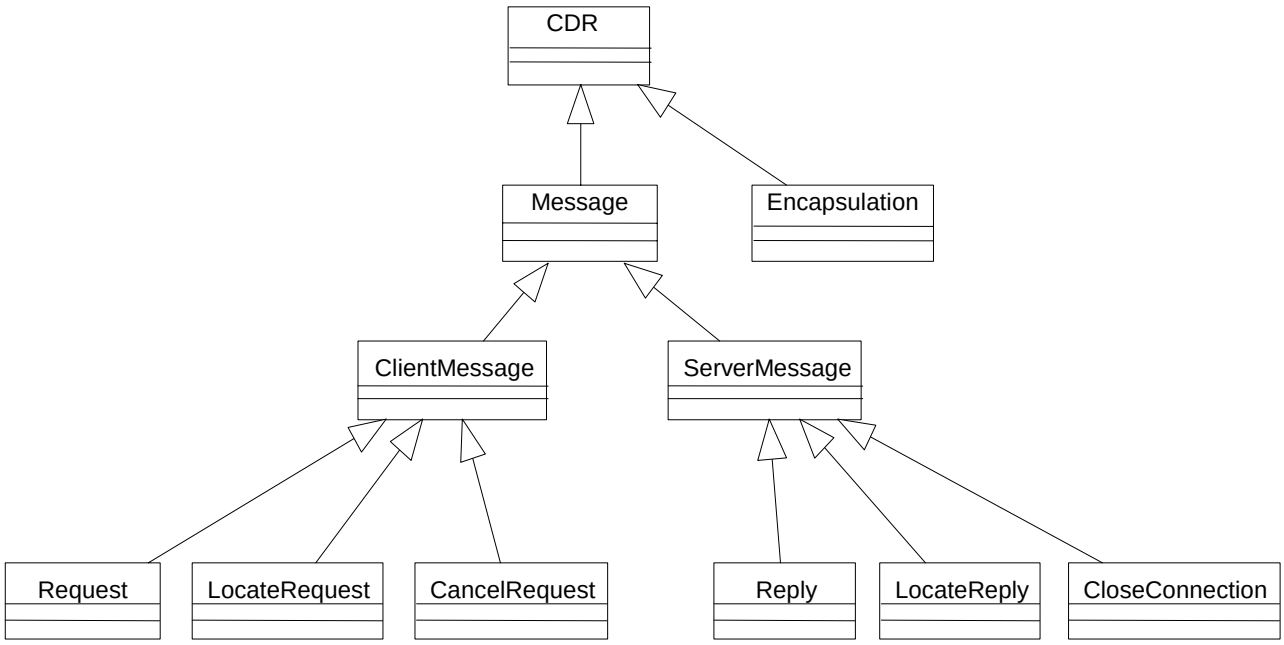

Figure 4-1 GIOP Message Representation Classes and Marshalling Classes

For example if the Java IIOP implementation was to create a request message coming from the client it would need to invoke the Request object constructor in the C++ API with the necessary parameters to create that object. A reference to that object would be returned to the Java side, which it could use to access the newly created C++ objects in the future. In the C++ implementation, the necessary creation of the client message, and message and CDR takes place.

Normally a native method calls a method and not a constructor. What was required was a way of creating an equivalent C++ object when a corresponding Java object was created. One possible approach called the *facade* pattern [Tines ‘99] was used to do this. With this pattern it was possible to conceal one object behind the other. The constructing instance of the Java class calls a JNI routine that creates a new C++ object, which returns a long to the calling Java

method. This long now acts as a pointer to the C++ object in the Java code by means of a long type. This should be a private final long so that it will never change (the JNI code can change its value) in the Java code. To invoke a method on that object referenced by the pointer, the long representing the C++ object is used and the relevant method called. In this way the Java becomes a proxy for the C++ class.

There were many objects in the IIOP API that required this, what could be termed, hack. This was a time consuming process. Since this did not add value to the goal of testing the D/IIOP layer disconnected functionality support. It was decided to suspend the integration with the ALICE framework and concentrate on the implementation of D/IIOP layer functionality using OBV.

4.5 Use Cases and Disconnection IDL

In order to develop some IDL that best represented the design goals for the D/IIOP layer it was decided to go through some uses cases for the key activities in providing this functionality. The Rational Rose UML tool was used to generate the use cases. The actors in the system were initially identified as

- The client application, this is essentially the user of the system that will require disconnected operation as a result of being on a mobile device, which at some stage will be disconnected from the network.
- The cache manager will look after the cache on the client side of the application, ensuring that the requested object is fetched and stored locally on the mobile host. It is also responsible for initiating the resolution process, by passing its stored objects back to the server side at a particular point in time.
- The server manager takes care of creating the copies of requested objects and passing them down to the client side. It is also responsible for conflict detection and resolution.

4.5.1 Connect

When the client side of the application starts up and needs to use server side functionality it connects to a server side object that offers that functionality. Thereafter, it connects to various server side objects depending on what it needs to do.

Figure 4-2 Connect Use Case

When the client passes a connect message through the D/IIOP layer, the IOR of the object that it is requesting to be connected to should be passed to the cache manager.

4.5.2 Using the Cache and Retrieving the Object

When the cache manager has a reference to the object the client application requires it needs to run a check on the cache to see if the object is already in the cache. It can also retrieve the object from the server side.

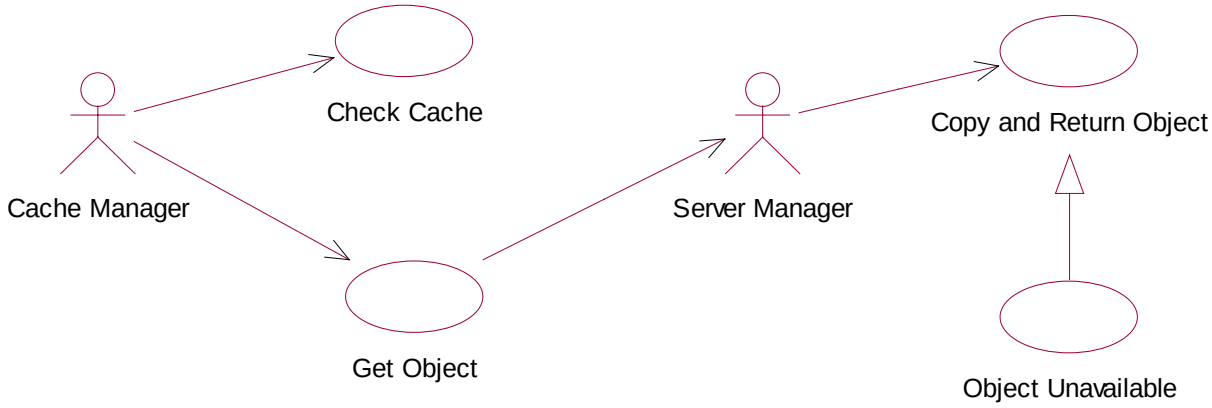

Figure 4-3 Retrieve Object

The server gets a request for the object and then makes a copy and passes the copy back to the client side. The server can also throw an exception if there are restrictions on making copies of the requested object.

4.5.3 Conflict Detection

The client is required to return the copies of the server side objects at a certain stage so that the server can perform conflict detection and if required resolution process on the objects.

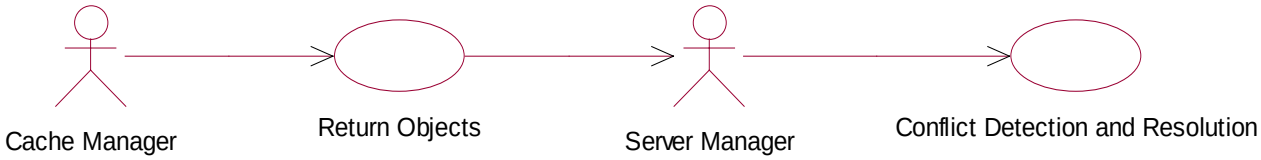

Figure 4-4 Conflict Detection and Resolution

4.5.4 Using Object-By-Value

Implementation of the use cases described in the previous sections is affected by the choice of OBV to pass the objects around. By using OBV the client initially connects to the remote server side object and then the cache manager retrieves the object on request, there is no analysis of the IOR. Future invocations on the downloaded object are not made through negotiation with the ORB or the D/IIOP layer. There is no IIOP message to be analysed instead the client directly invokes on the returned object. This meant that the implementation would not be compliant with the original abstract design for D/IIOP, however, it was decided to continue using OBV and use the implementation to study if it could be used to support disconnected operation.

4.5.5 IDL: Cache manager

It was originally intended to keep the caching mechanism generic so that any objects could be cached from the server and sent back to the server for resolution. The following shows the relevant IDL for the cache manager:

```
interface ManageCache {
      //1.
      //take a reference to the object you'd like to copy
      //pass back the value type to the client side
      ValueBase getValueType(in Object myObject);
      //2.
      //the client will want to ensure that changes
      //made to the local copy are passed
      //back to the server side at some stage
```

```
      void resolve(out ValueBase value);
};
```

The IDL had just two methods, one to get the valuetype from the server and one to pass it back to the server. In getting the valuetype, for a particular object, the reference to the object is passed as a parameter of the method. The server should then pass the valuetype associated with that CORBA object back to the client. The client can then use the newly downloaded valuetype directly. The process is reversed in the resolve method. The valuetype is passed up to the server and it begins the process of resolution.

It was found that when compiling this IDL the compiler had a problem with interpreting the ValueBase type. The generated code did not produce the stub correctly for the client side. It had no type on the return for the getValueType() method. It appeared that a specific mapping to a particular object was required so that the stub knew what object was going to be passed back. In other words the interface method had to be type specific and indicate which valuetype was being returned.

On inquiry at time of writing it was found that support for this type of IDL would be available in a newer release of the JavaORB product, version 2.1.1.

The process of defining IDL to pass the valuetype objects between the client and server had to be linked with the actual application. The next section describes the sample application and how its IDL was extended so that valuetypes could be used.

4.6 The Distributed Scheduler Application

One of the goals of the project was to implement a sample application that could use the disconnected support as specified in chapter 3. The sample application chosen was a distributed scheduler. This application was developed as part of an EURESCOM (http://www.eurescom.de/) project, P715 [P715 ‘99] It was implemented using OrbixWeb3.0 as a Java application that could be used over a telecommunications platform. The goal of this implementation was to port the application to JavaORB and to extend the IDL so that it supported valuetypes. In this way it would be possible to evaluate the necessary effort involved in enhancing an existing application in order to allow it to support disconnected operation.

The distributed scheduler was developed so that normal functionality of a calendar tool could be offered in a distributed environment. The application has most of the basic functionality of tools such as Microsoft Outlook and Netscape Calendar, however, it was defined using IDL thus enabling it to function in a distributed environment. The following class diagram illustrates the key components of the application.

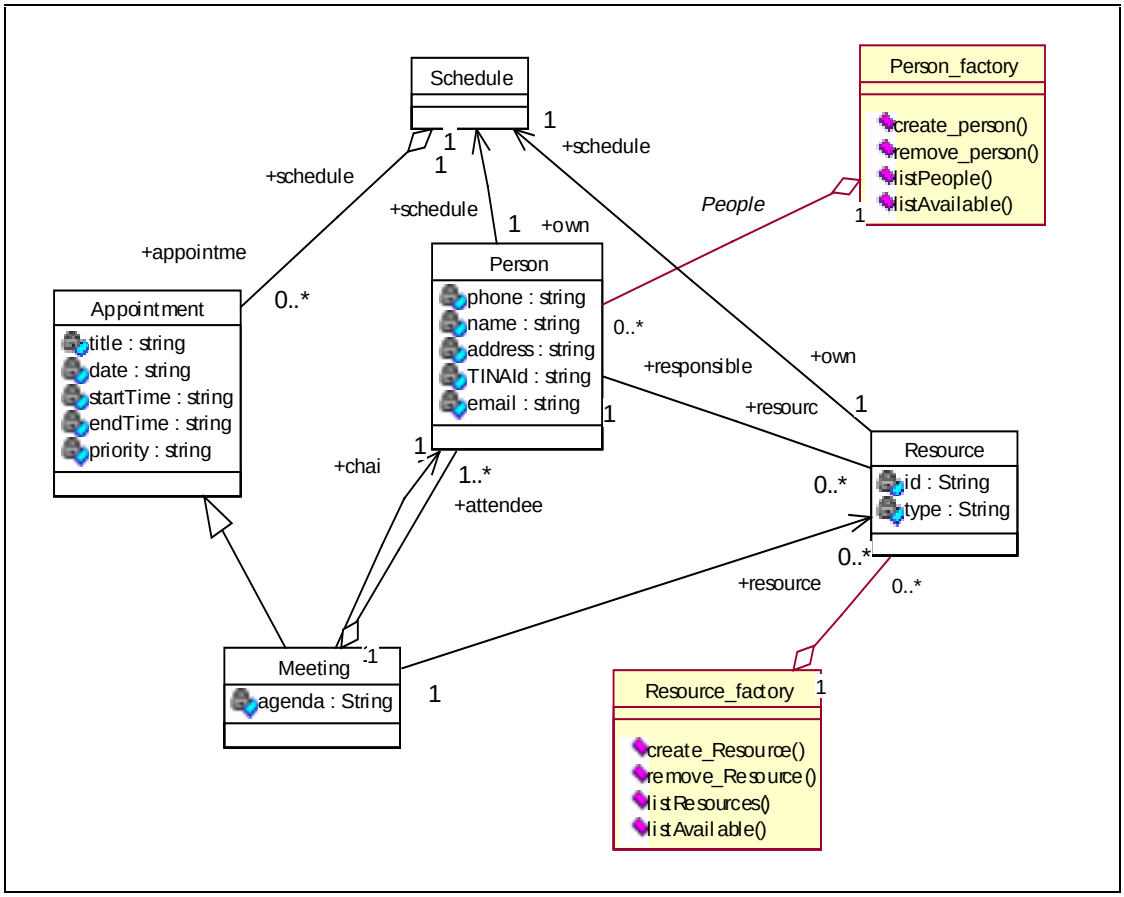

Figure 4-5 Derived Class Diagram Modelling Distributed Scheduler

- The central object in the system is schedule. Each person has a schedule where all the information relating to that person resides such as appointments and meetings. When a user starts up the system a schedule object is created for them.
- The person object basically contains all the information relating to a user of the system such as name, email address, location, telephone number. Each person has a schedule associated with him or her, so when a person object is created a schedule object is also automatically created.

- The appointment object contains all information relating to an appointment such as time, date, and location.
- Meeting is a specialised case of appointment where a list of other people using the system is associated with that appointment. The user can choose the attendees from a list of people registered with the application. Each attendee's schedule is be updated with details about this meeting.
- The resource object has details about various resources that may be associated with an appointment. These can include such items as a room or an overhead projector. A resource is chosen from a list of available resources in the same way a person is chosen for a meeting.
- There are two factory objects, resource factory and person factory, which are basically management objects. They create person objects and resource objects respectively and can then list these objects and finally delete the objects when required. The person factory is used by a management application for the system, which is used to add people to the system. Once a person is registered with the system an ID is associated with them. This ID is used when by the used when starting the scheduler application to indicate to the system which schedule object they are to be connected to.

The client development dealt with the user interface and functionality of the system, basically offering a standard interface which users would find easy to understand and intuitively offer the common functionality that is expected with all calendar and scheduling tools. Initially all the client functionality that adhered to the requirements of the scheduling application were implemented. This functionality was tested using a command line interface. The next phase was to integrate a Graphical User Interface (GUI) that would provide the standard look and feel of a scheduler application. GUI development can be a time consuming task so it was decided to employ a JavaBean™. A bean development company, *EnterpriseSoft*™ offered a calendar bean with suitable features to meet many of the basic requirements of the scheduler service. The front-end GUI code was then integrated with the client functionality specific to this implementation.

User interaction such as creating an appointment and viewing appointment requires remote invocations on the server object representing the user's schedule. By extending the application to support OBV it was intended to pass a copy of the schedule object to the client side of the application so that invocations would be local.

4.6.1 Integration of Valuetypes

In order to enhance the application so that the five core objects (Schedule, Person, Appointment, Meeting, and Resource) can be copied and passed around, each of the interfaces needed a valuetype associated with them.

State Variables **-** A requirement on the integration was that the valuetypes were expected to have the same state value as the CORBA objects. As a result when a client requested a download of a valuetype it would have the same value as the CORBA object. To enable this the CORBA object methods implementations invoked the valuetype methods where the actual value would be recorded in the valuetype's state variables. This ensured that the valuetype had a record of all the state values of the CORBA object. For instance for the Appointment object and valueAppointment valuetype the following IDL describes the attribute meeting title:

```
interface Appointment {
      attribute string title;
...
valuetype valueAppointment supports Appointment {
      public string vTitle;
...
```

In the implementation for the Appointment object the set accessor method for title is like this

```
public void title(String val){
      this.value.title(val);
}
```

this acts on the implementation of valueAppointment:

```
public void title(String val){
      vTitle = val;
}
```

Where vTitle corresponds to the state variable of the valuetype in the IDL.

Moving Schedule **-**The schedule object is the key object of the system and it is this object that initially needs to be moved to the client side to support disconnected operation. If this object is local to the client will be able to use the application functionality as normal. A valuetype called valueSchedule was defined for the schedule interface. The valueSchedule valuetype supported the schedule interface, which put a requirement on the valueSchedule implementation to

provide implementations for the methods defined in the schedule interface. The following is a section of the IDL for the schedule interface followed by IDL for valueSchedule valuetype:

```
interface Schedule {

    attribute AppointmentSeq alarms;
    attribute AppointmentSeq appointments;
    attribute GuiUpdate gui;

     void createAppointment(in string title,  in string date, in string location, in in string startTime, in string endTime, in string priority);
     void createMeeting(in People attendees, in Person chair, in ResourceSeq resources, in string agenda, in string title,in string date, in string startTime, in string endTime, in string priority) raises(Conflict);

...

     //this method passes back a valuetype version of the
     //schedule object
    valueSchedule getValueSchedule();

};


valuetype valueSchedule supports Schedule{

     //defining state variables for the value type
     public vAppointmentSeq vAlarms;
     public vAppointmentSeq vAppointments;
     public vGuiUpdate vGui;

     //just a test method to print appointments
     //use the state variable
     void printApp();
};
```

Behaviour modifications - In the original IDL for schedule the method createAppointment() returned an Appointment to the client. This had to be changed as a result of introducing a valuetype that supported the interface. Its implementation for the createAppointment() would

not be able to return an Appointment object because it would exist locally to the client and could not create the CORBA Appointment object in order to return it to the client. This had an implication on the server code in that the AppointmentSeq had to be retireved explicitly by the client to furnish the GUI with the values of the Appointment. The original schedule object was also changed to include the `getValueSchedule()` method which returned a copy of the valueSchedule valuetype.

Since the IDL keyword 'init' was not supported the instance of the CORBA object and the valuetype were closely linked. The keyword 'init' enables the creation of constructors for the valuetypes. It should have possible to leave the CORBA object implementations as they are and use another interface to create valuetype copies of the objects that are be passed to the client. When the CORBA object was created it creates its corresponding valuetype and subsequently supplies it with required data.

One of the methods in schedule was createMeeting(), which allowed the creation of a list of people representing the potential attendees at a meeting. These were to be notified by the server. Obviously this would not be able to happen if the user was creating a meeting on the copied local valuetype. This required an alternative way of creating a meeting by the client when the client was acting on the valueSchedule object i.e. when it was no longer accessing the original CORBA schedule object. Within the createMeeting() method a Meeting object is created the client is connected to schedule object a remote invocation. If the invocation is on valueschedule a Meeting valuetype is created, the other participants would be notified at a later point in time when the valuetype was uploaded. When the meeting is uploaded a CORBA object version of the meeting is created. This process is part of the resolve functionality that occurs at various intervals of operation, determined by the client. This conditional characteristic had to be implemented for a number of the operations on the Objects of the system, since the behaviour is different if the invocation is on the local version i.e. the valuetype or on the remote server side version i.e. the CORBA object.

The operation of the scheduling application was demonstrated, illustrating that the porting of the application was successful. Creation of a local appointment was also demonstrated illustrating that behaviour of the application could be the same when valuetypes were used in place of the remote object they were copying. With some further implementation it could be shown that the application could operate without being connected to the remote server side as long as it had downloaded the necessary valuetypes in advance.

4.6.2 ClassLoader

A class loader for the classes was also implemented. This was required by the application since it must had to be prepared to provide the implementations of the valuetypes to clients if the client could not find the classes locally. The class loader used a URL as the network location at which to find the classes it required if they were not initially found at its own location. If the classes could not be found at the URL an exception was thrown [Java ‘97].

Chapter 5

5. Evaluation

In this chapter the approach used in the project to achieve the original goals described in chapter 1 is evaluated. The chapter addresses how the concept of disconnected operation was brought into the CORBA world to improve availability of CORBA applications in the wireless domain. This is achieved through the definition of the D/IIOP layer and its inclusion in the ALICE framework. The definition of the D/IIOP layer is based on approaches used in a number of previous projects. The chapter addresses how the decision to choose OBV as the mechanism to achieve disconnection impacted on how far the implementation could meet the requirements of D/IIOP. This is followed by an evaluation of OBV and reflects how the focus of the project moved from strictly adhering to the D/IIOP abstract design to an evaluation of OBV. This was achieved by using OBV to enhance an existing scheduler application to allow it to work in a disconnected mode. A comparison is made between the D/IIOP approach and OBV approach in order to determine the most suitable approach to providing disconnected operation in the future.

5.1 Mobility in CORBA

In order to support disconnected operation it was necessary to find a mechanism that would allow server object replication. To keep the mobile client operating during disconnection, the server objects required to operate the client application had to be pre-fetched from the server on the fixed network and replicated in the client cache before disconnection occurred. The final phase of disconnected operation required the passing of the object back from the mobile client to the fixed network upon reconnection to the network. The process of conflict detection and reconnection would begin at this stage. D/IIOP was defined to support these mechanisms in the context of the ALICE framework.

A major design choice was the use of OBV to provide the functionality defined by the D/IIOP layer. The motivation behind the choice was that OBV was part of the CORBA 2.3 standard and appeared to be a way of passing copies of objects around in a way similar to the pass by value semantics of standard programming languages. In previous versions of the CORBA specification only the passing of references of objects allowed. The OBV valuetypes correspond to CORBA objects however they are not CORBA Objects i.e. they do not inherit

from CORBA::Object. This essentially means they can not be accessed in the same way as a CORBA Object, they do not have IORs that clients can use to invoke methods defined in the IDL for the valuetype. Even though OBV was not able to support all the requirements of the D/IIOP layer it was decided to proceed with its use since it appeared to offer a possible mechanism for supporting disconnected operation.

Valuetypes offer support for "data centric objects" whose nature is to be always passed by value (like a date, or a matrix). When the object is passed from the server to the client side, it leaves behind the capability of being accessed like a CORBA object. In the example code in section 4.6.1 the valuetype variable is accessed via the CORBA object accessor methods. When the copy of the valuetype is passed to the client and it exists as a local implementation on the client side. Any changes the client makes stay local and are not propagated back to the server side. The copied valuetype has its own identity and invocations never go over the network.

Another facet of using valuetypes that differs from using a CORBA object is that the client side needs to concern itself with getting the implementation for methods defined within valuetypes. Before clients could compile the IDL and start using the objects without being concerned with the implementation of the objects. Making the server responsible for passing code to the client is not feasible since it can not know in advance what implementation language the client side is using.

Downloading the valuetypes' implementations is quite straightforward for a Java implementation of OBV; the class loader can take care of that. However, since CORBA is used in networks with such a heterogeneous nature, and this will continue to develop in mobile networks, clients and servers could be written in different languages and they could execute on different architectures. It is not possible to download valuetypes in these scenarios. This increases the coupling between the client and server, which puts a constraint on the applications with OBV that a client can use.

OBV does offer an equivalent to passing a CORBA object by value. A client can access a copy of the CORBA object, which has the same state of the CORBA object at the time of the copy. However, transparency of the implementation is lost. The client has to have knowledge of the object's implementation before it can access it. It doesn't provide any support for replication so the tools such as conflict detection and resolution are application specific However, OBV does offer a way of implementing disconnected operation, but it must be accepted that this can never be transparent to the client.

5.2 Using JavaORB

The OBV specification is relatively new. Moreover it is still evolving. This has resulted in many ORB vendors not providing implementations for OBV. There were a number of requirements for choosing an ORB product in this project obviously it had to support OBV, and it had to be free. The decision to use JavaORB 2.0 from DOG was also driven by the fact it was a Java implementation, Java being more suited to the concept of passing objects around than other languages like C++. Section 4.2 discusses the motivational aspects behind this choice.

There were a number of problems relating to using the JavaORB IDL compiler:

- The ValueBase type that corresponds to the CORBA::Object type did not produce the correct stubs when passed through the IDL compiler. The ValueBase was a return type for one of the methods. This problems meant that it was not possible to follow the original design concepts of the D/IIOP layer making the implementation application specific.
- A method to get the corresponding valuetype for each object was defined in the IDL in place of the original approach. Due to a bug in the compiler the stubs it generated were incorrect and had to be changed by hand.
- It did not support the 'init' keyword that is used for initializers, so it was not possible to define the hooks for constructors for the valuetypes defined. This placed a constraint on the creation of the valuetype. The valuetype was created when the CORBA object (interface) it supported was created. Originally it had been planned for the server to create the valuetypes when the client requested an object copy. This approach would have fitted in with type of the server side support described in section 3.3.

These problems meant that it was not possible to follow the original design concepts of the D/IIOP layer making the implementation application specific.

The lack of documentation on the JavaORB product was also an inhibiting factor in its use. In its favour it was an ORB implementation that supported a large part of OBV the specification. There were some good examples of OBV IDL with the product that helped in gaining an initial understanding of how OBV worked. Its implementation of the naming service was easy to use and worked as expected. There was also a newsgroup [DOG '99] associated with the product where discussion in relation to problems with the ORB took place that was monitored by the developers.

5.3 Extending an Existing Application

This section describes some of the experiences that came from enhancing an application to support OBV. One important question is how does the implementation change due to the fact that different mechanisms are being used? The original IDL needs to be modified in order to incorporate the OBV functionality. The way in which the client operates changes as a result of the disconnected operation functionality. The server also needs modification to deal with the disconnected operation functionality. The following sub sections discuss these experiences.

5.3.1 Porting the Application

The application was originally implemented with Orbixweb3.0 and used the ORBacus name service. In porting it to JavaORB it was necessary to modify the way in which the application interacted with the ORB. When a CORBA object is created in an Orbixweb3.0 implementation the ORB automatically connects the object to the BOA and subsequently activates it. With JavaORB this process has to be done explicitly every time. Certain assumptions were made about one ORB based on experience from using another ORB causing problems with the implementation using the new ORB. This highlights one of the problems with BOA that motivated the OMG to define the POA. The POA provides features that allow applications and their implementations to be portable between different ORBs from different vendors.

On reflection it would have been better to port the application to use the POA especially since JavaORB had support for it. Any future work on the application would be independent of the ORB used.

5.3.2 IDL Enhancements

By choosing OBV major extensions had to be made to the original application IDL. Initially it was intended to develop a simple interface that just dealt with moving an object from the server to the client on request and from the client and back to the server at some point in time. This was not possible when using OBV. Each object that could be passed needed a valuetype defined in the IDL that supported that object's interface. The interfaces of each object that had a valuetype were also extended with a method that returned the valuetype associated with it. This is a significant change to an IDL specification that was initially intended to be simple. In using OBV there will be changes to the IDL particularly if there is a requirement for the CORBA objects to use the valuetypes defined.

5.3.3 OBV Application Enhancements

In using OBV there were obviously changes required in the implementation due to the changes in the IDL. An implementation was required for each valuetype. The implementations of existing interfaces needed to be changed in order to reflect the fact that they were associated with a valuetype. Since the keyword 'init' was not supported the instance of the CORBA object and the valuetype were closely linked. It should be possible to leave the CORBA object implementations as they are and use another interface to create valuetype copies of the objects that are be passed to the client. The keyword 'init' would enable the creation of constructors for the valuetypes. In this way the original application would not need to be altered, as was the case in this prototype.

These are obviously significant changes to an application just to make it support OBV. Some of these changes were due functionality of OBV not being available with JavaORB. The original IDL in section 4.5.5 would have require less manipulation of the original application code, however, it must be accepted that when extending and application for OBV support there will be alterations in the implementation.

5.3.4 Disconnected Operation Application Enhancements

The operation of the application was also altered to support the fact that disconnection was taking place. The application needed to know it was taking to a local version of the object rather than making a remote call to the server. This process could not be implemented transparently when using OBV. This is an application specific implication relating to the distributed scheduler. However it could be envisaged that this sort of outcome would occur in other applications when they are expected to operate in a disconnected mode.

There was a requirement on each valuetype implementation to be aware that it was to be used locally to the client. This did not affect most of the valuetypes as their operations were only related to data about the client using them. When a client creates a meeting object on the server a list of invitees is supplied and each of their schedules is updated with the meeting information. This is obviously not possible when the client is in disconnected mode. The client should be notified that the semantics of this operation have changed as a result of being disconnected. Only When the object is passed back to the server on reconnection is the normal operation of updating the invitees' schedules carried out.

The server side implementation requires a lot of modification if it is to support the mechanisms that enable disconnected operation. Initially there is the object copying and passing requirements. Then there are the requirements when the objects are passed back from the client, this may involve conflict detection and resolution as well as any changes that need to be made as a result of actions made by the client while in disconnected mode. These are significant changes in order to support disconnected operation.

The motivation behind using disconnected operation depends on how much more the client gets out of the increased application availability compared to the effort required in enhancing the application. The benefit of increased availability is hard to quantify. It is probably easier to determine the benefits on a per application basis.

5.4 Disconnected Operation in ALICE

One of the original goals of the project was to enhance the ALICE framework with a means of supporting disconnected operation. The D/IIOP layer was designed to provide this functionality. However, the decision to use OBV as the mechanism to allow for replication of the objects, which was one of the requirements of D/IIOP, meant that many of the original requirements for the D/IIOP layer were not supported in the implementation.

5.4.1 Transparency

A client using the D/IIOP layer uses IIOP in the normal way to connect to another object and to send messages. The D/IIOP layer intercepts that message and performs a fetch on the object required and stores a replica of it in a client cache. The client continues to use IIOP messages to communicate with the cached object. Ideally this communication would be transparent. However, this depends on the implementation of the D/IIOP functionality. If the IOR is translated transparently to the client and invocations forwarded to the new copy of the object, the client will still believe it is accessing the remote object. Invocations would appear to be on the same object. The D/IIOP API provided some functionality for fetching and resolving objects; this functionality could be provided transparently to the client so that the only API exposed by the D/IIOP layer is the IIOP API. However, achieving transparency depends very much on finding a suitable mechanism for passing copies of CORBA objects around the network.

For OBV, transparency is obviously affected. The client no longer invokes operations on the server objects using IIOP messages with the intervention of the ORB. The client accesses the

implementation directly and it is aware that invocations are local so it is also aware of being in a disconnected state.

5.4.2 Performance

It is not possible to quantify performance of the two approaches but certain assumptions relating to performance can be made based on the operation of the mechanisms. With D/IIOP, messages could to be marshalled and unmarshalled even if the invocation is local (the client believes the invocation is remote). This puts a computational overhead on the method invocation, which will take a longer time than a direct invocation on the implementation. With OBV calls are made directly on the implementation without ORB intervention, the process of building the IIOP message is by passed therefore it should take less time.

5.4.3 Server Implementation

Both approaches place requirements on the server implementations of the application. The server needs to provide support for creating copies of objects. It also needs to support a conflict detection and resolution process, so the changes in both cases are significant. It is impossible to quantify how much extra code is needed for D/IIOP in the way it was intended, however, for the OBV implementation of the distributed scheduler 20% more code was generated.

5.4.4 CORBA Compliant Operation

OBV was a fast solution to dealing with the issue of replicating CORBA objects. It did this at the expense of the CORBA functionality of the objects. Once the objects were replicated and stored at the client side the application operating using the replicas could not be termed a CORBA application. It no longer used the ORB or invoked methods that were exposed by the interfaces it was originally using. The application became specific to the implementation language, in the case of the distributed scheduler it became a Java application. It did however mean that a copy of the application was operational enabling disconnected operation.

Chapter 6

6. Conclusions

This chapter gives a summary of the work carried out during the course of this project and the resulting knowledge gained in relation to supporting disconnected operation in a CORBA environment. Finally, possible future work that could be carried out to give further insight into the domain is suggested.

6.1 Enabling Disconnected Operation

One of the achievements of the project was the proposal an enhancement to the ALICE framework by the addition of another layer that supported disconnected operation. The D/IIOP layer added further support for CORBA applications operating in a wireless environment. This enhancement brought the Distributed object computing mechanisms such as object replication and migration into the CORBA domain. Aspects of how CORBA applications worked at both the protocol-level and the application-level were altered because of the introduction of concepts behind the D/IIOP layer. The design proposed some changes to the standard client/server model in order to allow for disconnected operation. Some CORBA standard approaches were also investigated as a way of allowing this change in the standard client server model.

A phase of the project was to evaluate the D/IIOP layer through the implementation of a sample application, a distributed scheduler. The mechanism chosen to support the functionality defined by D/IIOP was OBV. Initially in the implementation a goal was to ensure that any mechanism could continue to use the ALICE implementation as it existed. Using Java as the implementation required the use of JNI to interoperate with the ALICE implementation. Using the JNI is a difficult and time consuming task so it was decided to concentrate on achieving one of the other original goals, supporting disconnected operation. The focus of the project turned to evaluating the feasibility of OBV supporting disconnected operation.

The distributed scheduler application was ported to the JavaORB implementation and its IDL enhanced to support OBV. This process illustrated to the shortcomings of OBV in terms of CORBA object replication and also the lack of portability of BOA implementations.

Many of the problems encountered in the project related to the fact that the OBV standard was immature with some of the concepts yet to be fully developed. The JavaORB implementation

of OBV was also incomplete, which lead to a number of changes in the original approach for enhancing the distributed scheduler to support OBV. OBV was found to provide the basic functionality to support disconnected operation for the application. However, it did require many changes in the original application before this could be achieved.

6.2 Future Work

Some further investigation of OBV could take place since new implementations are becoming available that are fully compliant with the specification. Both C++ and Java implementations will be available.

A CORBA compliant implementation of the D/IIOP layer that allows the movement of CORBA objects could be developed. The server objects could be replicated as full CORBA objects to the mobile host. The client application would continue to invoke methods on server objects as if they were on the remote server side. It would be very useful to analyse performance of the disconnected client compared to that of the connected client. It would then be possible to say if this functionality was feasible in terms of performance.

It would be useful to integrate the D/IIOP layer functionality with ALICE. This would require an extension of the Mobility Layer so that messages could be passed to the D/IIOP layer informing it of disonnections and re-connections. This would further enhance the support ALICE gives to mobile clients.

Another possible step would be to implement the conflict detection and resolution mechanisms required when server objects are passed back from the client to the server. It would be useful to investigate how the approaches used in standard distributed object projects would apply to the CORBA domain.

7. Bibliography

[Afonso ‘98] Ana Paula Afonso et al. UbiData: An adaptable Framework for Information Dissemination to Mobile Users, ECOOP ’98 Workshop on Mobility and Replication.

[Alkh '97] Hasan S. Alkhatib, Chase Bailey, Mario Gerla, James McCrae. Wireless Data Networks: Reaching the Extra Mile, IEEE Computer, December 1997

[Appenzeller '99] Guido Appenzeller et al. The Mobile People Architecture Stanford University Technical Report CSL-TR-99-777, January 1999

[Comer '98] Richard Comerford. Pocket Computers Ignite OS Battle, IEEE Spectrum May 1998

[Crow '97] Brian P. Crow, Indra Widjaja, Jeong Geun Kim, Prescott T. Sakai. IEEE 802.11 Wireless Local Area Networks, IEEE Communications Magazine September '97

[Cunningham '98] Raymond Cunningham. Architecture for Location Independent CORBA Environments, a dissertation in partial fulfillment of the requirements for the Degree of MSC in Computer Science University of Dublin, Trinity College Dublin.

[DOG ‘99] The Distributed Object Group Homepage http://dog.exoffice.com

[Emmerson '98] Bob Emmerson, 'On the move', Communications International April 1998

[Fiuczynski '95] Marc. E. Fiuczynski. A Programming Methodology for Disconnected Operation, University of Washington, ECOOP ’95.

[FT '99] Financial Times Media and Telecoms April '99 Editor Neil McCarthy

[Geier '98] Martin Geier et al. Support for Mobility and Replication in the AspectX Architecture, ECOOP '98 Workshop on Mobility and Replication.

[Haahr '99] Mads Haahr, Raymond Cunningham and Vinny Cahill. Supporting CORBA Applications in a Mobile Environment, Mobicom August 1999

[Haar '98] Jaap Haarsten. Bluetooth -- The Universal Radio Interface for ad hoc, Wireless Connectivity, The Telecommunications Technology Journal, No. 3 1998, Ericsson Review.

[Hild '95] Stephan G. Hild. Disconnected Operation for Wireless Nodes, ECOOP '95

[Inouye '97] Jon Inouye, Jim Binkley, Jonathan Walope. Dynamic Network Reconfiguration Support for Mobile Computers, MobiCom '97 Budapest, Hungary, Spetember 26-30, 1997

[Java '97] Java Network Programming, Ellitte Rusty Harold, O'Reilly

[JK '97] Anthony D. Joseph and M. Frans Kaashoek. Building Reliable Mobile-Aware Applications using the Rover Toolkit, ACM Wireless Networks October 1996

[JNI '98] Essential Java Native Interface, Rob Gordon, Prentice Hall

[Joseph '97] Anthony D. Joseph et al. Mobile Computing with the Rover Toolkit IEEE Transactions on Computers: Special Issue on Mobile Computing Janaury 1997,

[Katz '96] Randy H. Katz, Eric A. Brewer, Elan Amir, Hari Balakrishnan, Armando Fox, Steve Gribble, Todd Hodes, Daniel Jiang, Giao Thanh Nguyen, Venkata N. Padmanabhan, Mark Stemm. The Bay Area Research Wireless Access Network. Proceedings Spring COMPCON Conference 1996.

[Kistler '91] J. Kistler, and Satyanarayanan, M. Disconnected Operation in the Coda File System. Thirteenth ACM Symposium Operating Systems, Pacific Grove, US, 1991, vol. 25, pp.213-225

[Kojo '95] Markku Kojo, Timo Alanko, Mika Liljeberg and Kimmo Raatikainen. Enhanced Communication Services for Mobile TCP/IP Networking, University of Helsinki, Department of Science Series of Publications C, No. C-1995-15

[Lai '99] Kevin Lai and Mary Baker, "Measuring Bandwidth", Proceedings of IEEE INFOCOM '99, March 1999.

[LIFE '97] OMG LifeCycle Service Specification, OMG document: CORBAServices ftp://ftp.org.omg/pub/docs/

[MOJ '99] Programming Mobile Objects with Java, Jeff Nelson, Wiley.

[OBV '98] Object Management Group Objects By Value Specification, OMG Document: orbos:98-01-18 ftp://ftp.org.omg/pub/docs/

[OMG '98] CORBA Specification 1998. ftp://ftp.org.omg/pub/docs/

[OTDTF '98] Object Management Group Telecom Domain Task Force, White Paper on Wireless Access and Terminal Mobility in CORBA. OMG Document: telecom/98-11-09

[OTDTF '99] Object Management Group Telecom Domain Task Force, Wireless Access and Terminal Mobility Request for Proposals. OMG Document: telecom/99-04-02

[P715 '99] P715 EURESCOM Services Platform; http://www.eurescom.de/Public/Projects/P700-series/P715/p715.htm

[Palm '99] Palm VII Connected Organiser, Wireless Internet Access Comes to the Palm Computing Platform, 3COM White Paper

[Parrish '98] Chuck Parrish 'WAP zaps IT to wireless screens', Mobile Communications April 1998

[Phifer '98] Surfing the Web over Wireless, White Paper, A Core Competance Industry Report, January 1998.

[Pickar '99] It's a Wireless World, Marisa Pickar, Mobile Computing May 1999.

[POA '98] OMG CORBA V2.2 The Portable Object Adaptor February 1998 ftp://ftp.org.omg/pub/docs/

[Raatikainen '97] Raatikainen et al. Service Machine Development for an Open Long-Term Mobile and Fixed Network Environment. 1997 DOLMEN Consortium. ACTS Ref:AC036 DOLMEN

[Redl] Siegmund H. Redl, Mathias K. Weber, Malcolm W. Oliphant. An Introdcution to GSM, Artech House

[Regan '97] Regan, Bell Labs Technology: Trends and Developments, April 1997.

[Saty '89] Satyanarayanan M. Coda: A highly available file system for a distributed workstation environment, Second IEEE Workshop on workstation operating Systems, Pacific Grove, US, September '89.

[Silber '99] Steve Silberman 'Just Say Nokia', Wired September 1999

[Sorensen '96] Sorensen, M.G A mobility-transparent model for Consistency, Technical Report DIKU-96-3-7, University of Copenhagen, Dec '96.

[Tines ‘99] Mr Tines at Newsgroup: comp.lang.java.programmer April 1999

[UP '98] The Wireless Application Protocol, Wireless Internet Today, White Paper, November 1998, Unwired Planet

[Vinoski ‘98] Steve Vinoski. New Features for CORBA 3.0, Communications of the ACM, Vol. 41, No. 10, October 1998.