Volume 2 No.4, APRIL 2011 ISSN 2079-8407
Journal of Emerging Trends in Computing and Information Sciences

http://www.cisjournal.org

Employing CORBA in the Mobile Telecommunications Sector and the Application of Component Oriented Programming Techniques

Paul Storey[#1], Raul Valverde[*2]
[#]University of Liverpool, UK
[1] Paul.storey@spiess.co.uk
[*]Concordia University
John Molson School of Business, Canada
[2]rvalverde@jmsb.concordia.ca

ABSTRACT

This paper focuses on the implications of employing CORBA Middleware in today's wireless telecommunications services and applications developments. It seeks to understand what current opportunities exist to support the use of this technology and assesses some of the performance implications that might be encountered. Discussion also revolves around the use of Component Oriented Development techniques in this same vertical and observations are made as to the benefits and practical problems associated with pursuit of this methodology in wireless telecommunications development programmes.

Interest in this topic was aroused after observing the continued development of wireless telecommunications technology (Third Generation - 3G) that allows users to maintain permanent but connectionless contact with more traditional Wide Area Network (WAN) & Local Area Network (LAN) based services. As the boundaries between cellular, wireless LAN and traditional fixed network technologies become increasingly blurred, trends are emerging that seek to extend and migrate the technologies that are currently employed on cabled and wireless LAN networks over to the cellular networks and mass market user equipment.

Keywords: *3G, Component, CORBA, ORB, Performance, Telecommunications, Wireless.*

1. INTRODUCTION

Industry uses of Middleware can be found in all the following vertical markets to one degree or another:

- Technology
- Telecommunications
- Transport & logistics
- Travel & tourism
- Financial services
- Healthcare
- Infotainment
- Insurance
- Manufacturing

There are many more besides, but the above gives an idea of the pervasiveness of Middleware today. Wherever software is utilised in the above verticals evidence of Middleware can often be found, and in many cases bespoke software that might have been developed for and aimed at a particular vertical market can in theory be used across other verticals. The spread across the vertical markets of a particular software application is widely referred to as 'horizontal software' because of its span. The mobile telecommunications vertical is one area that is yet to receive this technological broadening through the medium of middleware in any significant way and is probably due to several restricting factors that will be discussed.

Common forms of horizontal market software like email, word processors and spreadsheet programs that have migrated onto wireless mobile platforms do not typically employ Middleware technology, but follow traditional routes for their development and deployment. The perception that middleware might be being employed in the case of two unrelated client/OS/platform application implementations is understandable, but these connected wireless applications probably just conform to the standard interface defined for the server that handles data exchange at an appropriate layer.

Many definitions of the term Middleware are on offer and it is very clear that the term Middleware means different things to different people. One concise definition that the term Middleware covers is "the layers of software between client and server processes that deliver extra functionality and that hide the complexity of that extra functionality behind a common set of APIs that client and server processes can invoke" [1]. Middleware when viewed in this context sounds like a pretty simple set of software libraries that can be used by any developer as a simple way of creating applications that can interact, without the developer having to worry about the underlying detail. From a management perspective having a mechanism where the software developers can implement applications without creating a multitude of proprietary interfaces and creating interwoven legacy code is good news. Freed from such costs attaching themselves to custom implementations and add to that the large reductions in future maintenance expenditure then Middleware seems to be an ideal development platform and would especially benefit the wireless mobile

application development community. Having said that though, applications that utilize traditional underpinnings (application/OS direct interaction) versus those that communicate through the Middleware layer will behave very differently at the non-user interface level (i.e. the lower Middleware layers) and hence benefit the developers and maintainers directly. To the user there may be no difference, and in the case where a legacy application has been ported to operate on Middleware this would be a classic example. For the developers though, a decision has to be made as to how best to support their users with the Middleware technologies on offer. One fundamental question that narrows the focus on the category of Middleware that might be employed in the wireless mobile application domain surrounds the characteristics of the Middleware technologies available and the fit to the needs. Some of the different forms of Middleware are captured below:

- Component Object Model (COM / Distributed COM)
- Distributed Computing Environment (DCE).
- Message Oriented Middleware (MOM).
- Object Request Brokers (ORB).
- Remote Data Access (RDA).
- Remote Procedure Calls (RPC).
- Transaction Processing (TP) or Distributed TP (DTP).

Latter day Middleware frameworks such as CORBA support the component oriented development programs that address the issue of the enterprises desire to utilise commercial off the shelf (COTS) software products that are ready proven on diverse infrastructure elements [2]. Breaking large projects down into a series of smaller, lower risk initiatives is not new, but when these can be integrated onto a proven middleware platform such as CORBA it can reduce still further the chances of project slippage and/or cost increases. This component oriented and ORB based approach is gradually feeding into the mobile telecommunications sector as increasing numbers of UE manufacturers can and do purchase COTS protocol stack software for integration into their mobile telephony product offerings and settle on common Operating Systems. At the present time though these components still require a high degree of effort at the point of integration as no UE/PDA hosts presently support Middleware. Increased technology convergence and standardisation in the mobile telecommunications arena have lead to fewer tools, platforms, and methods being used than traditionally were in the past. Development methods and modeling notations such as UML/SDL when coupled with automated code generation tools are facilitating the movement away from the dozens of different design methodologies, programming languages and operating systems.

As all mobile telecommunications systems have inherent real-time considerations to be met, the asynchronous remote procedure call (RPC) and message-oriented middleware (MOM) variants of Middleware were never going to make it to the top of any list to chose from when it comes to adopting one technology or another. Other forms of Middleware such as SQL-oriented / Remote Database Access (RDA), is also ruled out due to its unique positioning for pure database oriented developments. DCE, Synchronous RPC and ORB Middleware frameworks remain [3,4] and the focus of this paper on CORBA was made in part on the grounds that DCE and Synchronous RPC Middleware are already employed to a large extent in the telecommunications vertical. Equally important in this respect is that although widely used, these technologies will probably not make the same impact as CORBA could if adopted. Although DCE & RPC provide an environment in which many solid benefits can be derived they cannot meet many of the desired objectives that facilitate true component oriented development programs and the referencing of static services.

CORBA and DCE both have their roots in Remote Procedure Calls and are a standard part of the GNU C libraries that are widely used in wireless mobile device and infrastructure developments. Because RPC is very commonly used middleware application, extending RPCs is an entirely valid approach but CORBA supports self-describing objects that can discover each other, query interfaces and make method calls without any compile-time knowledge [5]. This ability to discover objects ensures that CORBA is more powerful than RPC elements that have to invoke fixed methods.

2. MOBILE TELECOMMUNICATIONS ORB ARCHITECTURE SELECTIONS

2.1 ORB Requirements

One key element in making a successful ORB offering to the embedded wireless telecoms sector would arise out of being able perform two things.

Firstly to be able to cover all of the Operating Systems in use on the UE's and PDA's today:

- Palm's OS
- Windows CE & Pocket PC 2002 OS
- Symbian's EPOC OS
- Embedded Linux OS (approx. 15 variants)

And secondly to provide appropriate device control or support of between 15% and 25% of the OEM's platform offerings detailed below:

- Alcatel
- Casio
- Compaq
- Handspring
- Hewlett Packard
- IBM
- Ericsson
- LexiBook
- Motorola
- NEC
- Nokia
- Palm
- Panasonic
- Philips

Once ORB's are available on this proportion of platforms and the OS's, then penetration into the developer community should be sufficient to allow smaller developers to start implementing applications for them. Many OEM's now use similar, if not identical chipset's and re-badging is common place so the range of device specific variants to cover a larger manufacturer base is easily possible.

Some degree of showcasing would need to be done to spark the interest, but as the take up of CORBA technology grows still further and applications migrate out of the traditional domains adoption after that point should be swift providing any specialist development libraries on offer were free [7]. The next few sections identify areas on the ORB selection process that require that real time OS and real time application developments are not compromised by the inclusion of an ORB. Later this paper looks briefly at the ORB selection criteria and how it must also fit with OS criteria concluding that ultimately an unmatched ORB/OS pair is bad. This gives rise to a bit of a 'catch-22' situation arising, as in doing this pairing, CORBA services risk getting welded into a particular operating system that again can then directly compromise the platform independence of the ORB.

IIOP is the communication protocol that all ORB's support as a minimum. This mandatory ORB interoperability protocol allows ORB's to communicate with each other in a way that is seamless to the client and servants when remote invocations are made. This mechanism is great in that it allows mobile users to make service requests without regard to their location, but the underlying connection management within the ORB has to manage the requests made and notify the client applications if problems occur [5]. In an ORB that is specifically designed for the wireless UE platform connection management in the ORB is crucial. The ORB would have to be able to distinguish between a temporary loss of service due to packet loss through signal degradation or IP infrastructure network congestion and between the permanent loss of communication through total connection failure. Temporary loss of service detection would require some kind of configurable hysterisis timing mechanism before shutting down the connection, and permanent loss of communication through total connection failure notification from the wireless protocol layer might lead to the ORB attempting to re-establish the connection without notification to the client. Take another practical but realistic scenario of a user initiating multiple sessions that involve the creation of an equal number (or more) client connections to be established. In this case the ORB would need to be able to schedule access to the services requested in priority order. For example if ten clients are operating on the UE and 6 of them are idle then the connection manager would need to arbitrate and close the idle connections to make better use of the connection space available for the other clients requiring service. Once complete the previously idle connection could be re-opened. However, repeated attempts at re-establishing a connection would also need to be configurable. If not, then the ORB would be just as wasteful of resource when trying to gain service when it is not possible to grant it at a wireless protocol level as it would be to leave a connection open when it has been left unused for a considerable time.

All ORB's provide a suite of API's to the applications that utilize their services and the same is true for the ORB to OS interface. Each ORB implementation has to be built for a native platform and Operating System to allow the use of the OS services on offer. An ORB's lower layer interface or 'OS adaptation layer' needs to encapsulate the OS API's for multithreading and task or process synchronization. The support of multithreading in real time wireless embedded applications is essential to allow concurrent tasks to execute simultaneously. For example if a UE device were to connect to a PDA using its USB, Bluetooth or infrared port for data transfer and the UE device has to also signal to the base station or Node B using the air-interface protocol of choice (GSM or UMTS). In this fundamental mode of operation many tasks would be involved, all required to operate concurrently to provide the service. As such all real time OS provide support for this and compatible ORB's need to extend this up to the application layer. Many real time operating systems support POSIX compliant Pthreads, however being able to rebuild the ORB so that it can employ LINUX threads and Win32 threads is essential if the ORB can roll out across the wide range of OS's that run on today's PDA's. Many UE devices use less prolific Operating Systems like Lynx, OSE, Tornado, etc. and ORB support for these are considered essential if the mobile community will obtain the full benefit. The difficulty that is observed is how do the ORB's across these platforms mask the underlying differences that exist in the scheduling and task synchronization of the different OS regimes. Depending upon the applications implemented on an ORB for the UE, a situation could then exist where the scheduling mechanism used on one platform differs from that used on another. There may be differences also in the efficiency of the context switches that a particular OS can support and this will impact on the multithreading and inter-task communication aspects. From such minor differences in performance the behavior of the ORB could be affected and ultimately the ORB applications will be impacted. It has to be said even if this is to a lesser degree, that the dynamics will be changed. It is essential therefore that the task priorities at the OS level can be mapped onto the object application priorities to preserve the dynamics of the system.

Just like multithreading and synchronization above the ORB adaptation layer has to encapsulate OS APIs for synchronous and asynchronous demultiplexing of input/output, timer, signal, and synchronization based events. Like their LAN node counterparts, ORB's that would run on embedded wireless devices also need to dispatch application specific handlers in response to the various types of events. Across these Operating Systems the ORB will require support for event handlers like the select(), poll(), and Win32 WaitForMultipleObjects facilities along with the mapping of process or communication signal events. In any case, predictable and

well defined event handling is a priority, but in embedded systems event handling and the ORB's robustness is much more critical. Depending upon the individual ORB support for the mapping of OS events to CORBA events and the ability of the application developer to query and action any exceptions raised through to it, then the different event handling regimes in the ORB's will lead them to being more useful in critical real-time systems. If one ORB can accurately map and react to all OS events that might be raised then the system is less likely to fail as a result of unhandled events or exceptions. This quality is essential in embedded systems where the effects of a hanging application or crash because of an unmapped event could be catastrophic. In a UE context where a single user is affected the impact is probably just an annoyance, but in the contexts of a Node'B' or RNC the result could be several hundred lost connections some of which may be carrying an emergency call. The same goes for the applications implemented to run on the ORB, if best practice is being followed the client application developers will be building in the same exception and event handling mechanisms that are offered through the ORB.

Although Fault Tolerance is a desirable property in many systems, when it comes to ORB implementation on real time embedded devices like UE elements then the overriding concern is not to tolerate faults, but to have the handling mechanisms and recovery systems ready built in to determine what corrective action to take (see section above). Allowing for inter-orb or inter-application messages to go missing and/or not binding synchronous events by timers et.al will inevitably lead to missed deadlines and hence the departure from fulfilling the constraints of what should be a deterministic real time system. Because wireless telecommunications requires a high degree of timing accuracy for the transmission and reception of signaling between the UE and base stations then a fault tolerant ORB in these areas is not going to be required even if the fault is transient. Sometimes in critical wireless telecommunications infrastructure elements the applications that run on these nodes are duplicated on co-located hardware and cross connected so the applications can perform hot, warm or cold changeover in the case of failure. The fail over mechanism is usually application specific, but support of this feature through the use of an ORB makes the facility much more attractive. A primary or master unit that has an ORB running the mission critical applications could be set up to communicate transparently with its standby or secondary unit with the same ORB/application deployed on it. This facility then frees the application developer from the need to build in the communication mechanism between the live and redundant element. In the case of the infrastructure element failing, the remote UE client will not (or doesn't need to know) if the services it is receiving through the ORB are coming from a primary or a secondary (standby) server.

Fault detecting like Fault tolerant ORB's will be desirable in many areas other than embedded real time systems. Not that fault detection would be necessarily bad in UE elements where a self aware ORB might decide when to initiate recovery (like a reboot), but in wireless infrastructure elements the detection of a fault needs to be passed up and handled at the application layer. This way the application itself makes the choice of what corrective action to take based on designed in behavior (real time deterministic behavior). If the ability to detect and correct faults at the ORB level were inherent, then without application specific knowledge the reaction to a detected fault could again be catastrophic. Graceful failure is desirable in wireless infrastructure so that application threshold monitors can alert other (external) systems that problems have been observed. If air interface resources in UMTS or a GSM systems are degrading the loss in resource capacity at a base station would need to be alerted to an operator in a position of control. If the ORB were to take it upon itself to reboot the system when the situation becomes uncorrectable this external control and predictability is lost. In this way the monitor application (be it human or otherwise) retains control for the correction of application affecting faults after they have been detected. There could be circumstances where the UE/PDA applications might be affected from lack of system resources and it is imperative that these faults, when detected, are passed back by the ORB and handled at the user or application level (i.e. out of memory and selective application closing by the user).

Because use of an ORB abstracts the application developer from the underlying communication mechanisms the issue of controlling the quality of service (QoS) from end-to-end becomes an extremely important element in ORB selection criteria. To ensure that an ORB can provide adequate support for real time embedded applications like nearly all those found in the wireless telecoms vertical then the following issues need to be considered carefully.

Primarily the opening up of communications channels between UE clients as infrastructure based services through an ORB needs quite tight coupling. Any delay between making a request for service and the response being received from the remote servant will largely be influenced by the latency in establishing the underlying communications path. Once the communications path is open, then the application may then need to raise the bandwidth required, or the condition of the network and any congestion being currently encountered may need to limit the bandwidth available. In the real-time embedded wireless domain then the QoS of an ORB is directly related to its ability to control or reflect the underlying transport layers [8]. Considering then the newer IP based offerings on 3G systems like Push To Talk (PTT) services that operate on top of Voice Over IP (VoIP) data services, this latency and QoS characteristic would need very careful consideration.

Being able to apportion the degree of latency introduced in any system by any one element is key to being able to assess its suitability for an application. By introducing an ORB into the communications framework this issue is further clouded, as the impact of simply adding one will throw out of balance any optimized

application, operating system and protocol stack trio. As already stated for real-time ORB selections one of the keys to selecting a suitable ORB is how it maps the OS API up through to the ORB API for the application developer to make use of in meeting particular QoS requirements. Good or suitable ORB's will allow application control over the scheduling and mapping of client priorities onto concurrent threads to reduce latency or jitter (National Laboratory for Applied Network Research, 2002). Even if the ORB selection criteria is met and a suitable embedded real time ORB is utilised it is crucial that the underlying OS is supportive of the real time considerations as one without the other is equally damaging. What might be perceived as ORB latency, throughput and Jitter may equally well being the OS being used as could a badly designed or implemented application.

2.2 Other ORB Requirements

Because of all the factors described above ORB selection has to be considered very carefully. In extolling the virtues of using ORB's across the embedded wireless communications market it is clear that unconditional selection is simply not possible and each platform/OS/application use whether it be on the UE, a PDA, a Node'B', or anywhere else in the UMTS infrastructure.

For developments in telecoms Middleware to be furthered, other factors have to be considered. The general principle that centers the ORB round central serving nodes within the enterprise/ infrastructure needs addressing and the focus of some ORB development to be optimized around the User Equipment. In optimizing the UE centric ORB's then vendors of these will also need to consider pricing and licensing. Due to the economies of scale large volumes of UE's are manufactured and manufacturers are unlikely to want to pay for the licensing of these specialist user centric ORB's.

Many established licensing models exist and some UE equipment manufacturers have arrangements with OS vendors to supply OS target licenses free of charge on the handset platforms and in return the royalties normally payable are bought outright or recouped from the developer tools required to support application development for those OS's provided. If ORB licensing is a model that is used then the UE market penetration will be decided not only on technical suitability but also on price and licensing models available. ORB licensing costs across the infrastructure could be absorbed easily into the sale price of these high value nodes (RNC's, Base Stations, Node B's etc.), however the license costs of ORB's on UE or PDA equipment when multiplied up over the millions of units on the market would need a unique proposition so as not to slow the adoption rate.

ORB vendors could follow the same model, or they might obtain their return by offering consultancy services for ORB migration to OS's and platforms. Another model that might be considered is the supply of application libraries to ORB application developers that they can re-work and re-sell components and offer yet greater reductions in time-to-market for such developments.

All these factors and more are considerations in the selection of ORB's for such and maybe the vendors have to make money off the services they offer and partner the network operators.

Many CORBA implementations are Open Source and large proportions are C++ based. By allowing modification and configuration of the ORB's the developers that use them have more power and greater control over how they chose to utilize the services and create light weight or efficient ORB's. As a result of the ORB's being open source a broad-based user community exists that can be drawn upon. It is recognised by many of the larger commercial ORB vendors that by offering good support they can ensure their offerings are used on large developments by other large organizations that rely on backup.

3. CORBA IMPACT ON 3G SERVICES AND DEVELOPMENTS

3.1 Practical ORB Overlay on 3G

Supposing all the real time ORB requirements in section 2 are bettered and met respectively, then full deployment across the 3G infrastructure and beyond could become a reality sooner than expected. In providing services to the UE using a CORBA framework then one consideration to be made is in the host location of the serving elements. Although location independence is a keystone of the ORB applications some practical aspects need to be addressed when dealing with highly agile client ORB objects.

In a 3G network architecture, the Node 'B' element is the direct air interface component and is responsible for setting up and managing the logical channels over the air interface. This could be one logical place to host ORB services serving the UE's.

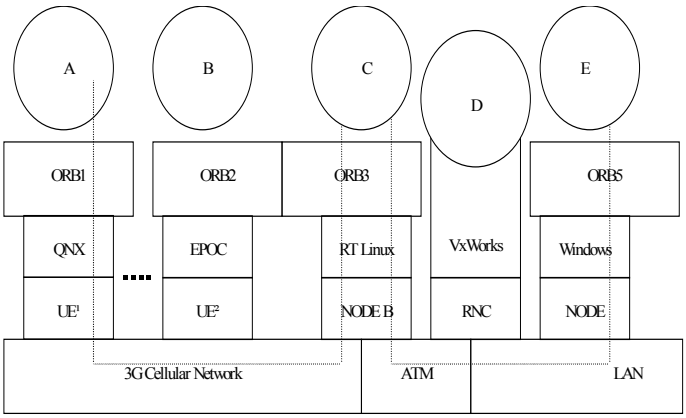

Figure 3-1 – Node 'B' Hosted ORB

But, in terms of its functionality, the specification and definition of the Node 'B' in the 3G series of

specifications does not cater for hosting such services. The Node'B' essentially takes responsibility for opening the communications path from the UE through to the target element wherever that might be. The next element in the infrastructures core that could support the hosting of such services would be the Radio Network Controller (RNC). In effect then, along side each RNC node would sit a UE services host that could provide CORBA services transparently to the UE nodes by breaking out the appropriate ORB communication messages that are embedded within the UMTS protocol.

Depending on the traffic profile and more importantly the UE initiated CORBA client message volume coming through the RNC to the RNC hosted CORBA services, the ORB that is selected would have to be appropriately scoped. A bottleneck is likely to occur if the CORBA services on offer at this point in the network were not controlled and balanced accordingly. As such the ORB bandwidth reflects directly on the service response time that the user expects which may degrade during busy periods or in high density urban cell sites.

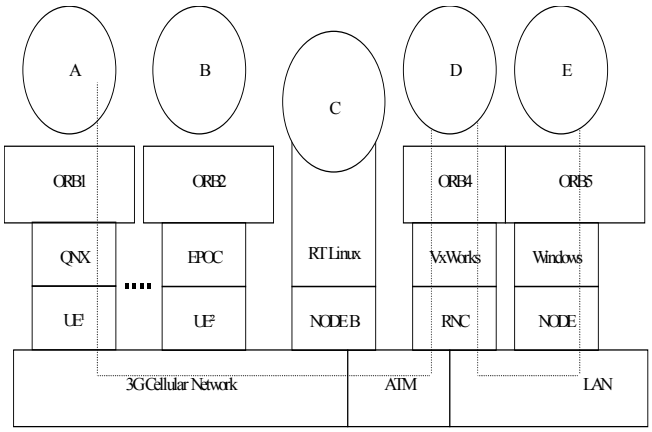

Figure 3-2 – RNC Hosted ORB

In both the above cases the user could access the CORBA hosted services transparently using standard connection management methods built into the ORB. As the UE moved to a new cell the ORB could be notified by the underlying protocol stack and be provided with the location of the new naming server and follow this up with a locally routed service request. Once communication had been established the CORBA hosted application clients could make requests of the services on offer.

One major consideration that would need to be made in either scenario is the how the inter-ORB communication between the individual RNC hosted ORB services are handled. If the ORB host were at the Node 'B' or RNC then CORBA service hand over would need to be synchronised. Although the ORBs hosted at these two points could make use of the LOCATION_FORWARD mechanism, the frequency and the time between service hand over could be often as the UE moves from one adjacent cell (and maybe back again) in a very short space of time. The net effect of this is to place additional strain on the Node 'B' and RNC elements and then on the hosted ORB's also in the shutting down and re-starting of servants on adjacent ORB's. A more suitable position for the ORB hosted UE services would be deeper in the infrastructure past the SGSN (Serving GPRS Support Node) and GGSN (Gateway GPRS Support Node) elements so as not to pollute the operation of these fundamental entities.

However, as the ORB is deeper within the infrastructure the user base needing to then gain access to the central point will rise and the concentration on the requests for ORB hosted services could become excessive.

Other than client side mobility giving rise to servant re-location shown in

Figure **3-1** &

Figure **3-2 – RNC Hosted ORB**

, another factor to consider if the scenario shown in

Figure 3 -**3** is employed is the issue of load balancing. As discussed in the previous paragraph if there is a central host of the CORBA based UE service offerings then clearly the work to be done at any one particular time on that server could be huge.

Regardless of the services on offer to the UE ORB based clients the same technology that is currently in use now in support of CORBA services load balancing can be directly employed. From this point on in the infrastructure the services on offer to other CORBA client entities is absolutely transparent.

Within a not too excessive time frame there could come a point in the future where the UE originated client requests would be routed using higher layer (layer 4/5) switching technologies to a users home PC that might host its own ORB servants. This scenario would then remove the need for service provider hosted servant objects for all but the lucrative revenue generating services the personal user could not host on his 'Personal ORB Framework' (POF).

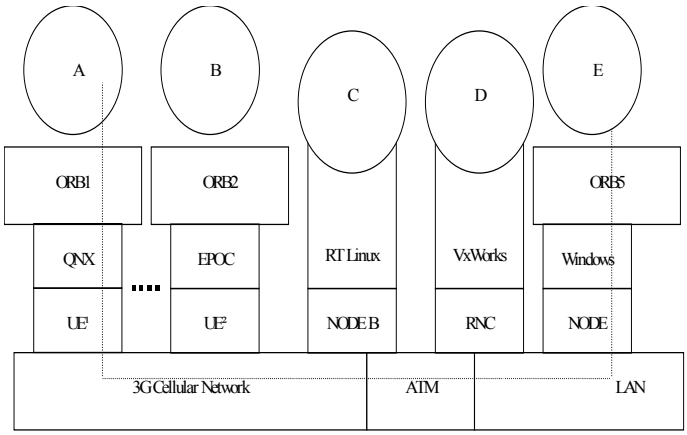

Figure 3 -3 – Deep Infrastructure Hosted ORB

3.2 Applications & Services

Title must be in 24 pt Regular font. Author name must be in 11 pt Regular font. Author affiliation must be in 10 pt Italic. Email address must be in 9 pt Courier

Regular font. The example below attempts to highlight some of the variations in applications, Operating Systems and Hardware that is present in the existing GSM wireless network. The same technology mix has been preserved and carried forward into the next generation mobile network (3G) and has been largely due to the need to provide a migration path for the user community until the new technology can be relied upon. At the time of writing whether on GSM or the imminent 3G networks if application 'A' & 'E' need to communicate, each has to be developed for the specific OS it runs on and in some cases for Operating Systems that are highly optimized for the platform.

What we then have is the archetypal client server relationship with the UE fulfilling the role of the client and the Node'B', RNC or elements deeper within the infrastructure fulfilling the role of the server. Even for the much simpler case where a UE application, say an appointment book is required to be synchronized with a directly connected host (PC or laptop) the applications for each party in the relationship have to be developed with a single platform in mind. Because of the huge variety of computing platforms contained within the telecommunications sector, and in that user equipment (UE) is included, any application provider has to target the development to a specific platform pairing, or spread their valuable developer resource across the available platforms.

Compound this pressure with the time to market of the application and the sheer number of platforms, even if the provider is a global player the cost of development and deployment across the board will always be high. To reiterate, the non-exhaustive list of platforms that are available number over 20 and the number of operating systems supported across this range of platforms has grown significantly with the advent of embedded Linux kernels into this market segment to over 15. The Symbian alliance formed between some of the OEM's leads to a narrowing of the OS scope, but even these platforms that were EPOC native now have Linux OS's migrating on to them. Other traditionally embedded Windows based platform such as Sharp, Toshiba, Casio and Compaq devices also have Linux ports. This all provides for massive consumer choice but the choice is traded off against proprietary horizontal applications that each vendor wants to offer and a seemingly ever increasing number of protocols and data exchange formats to keep up.

By taking a sniper approach, SME's can create proprietary applications for a platform pair (e.g. A&E) based on projected uptake of the platform and the application but this model is highly risky. The same SME's could also produce applications for the platforms shortly after launch and hope to catch the tailcoats of the next market leader. Despite a relatively buoyant mobile services market place in comparison to the general downturn in the telecoms market, the client/server applications that might present themselves as the real killer applications have not materialized. Sure enough UE applications written for an operating system such as Symbian's can run on any UE or PDA that employs it. Users and developers derive benefits from this approach by maintaining consistent usability and gaining reusability respectively, but this is still within one platform/OS envelope. For such applications to break out of the straight jackets that tie them the ORB vendors need to create the opportunity.

Middleware that is widely used in places such as tracking/logistics, data gathering / utilities is seeping out from within the corporate infrastructure and is now emerging as user services at the customer interface. This is apparent as users have the ability to query (through a web site) the Middleware enabled backend systems of parcel tracking systems. For the case in point (parcel tracking) extending the Middleware framework to cover the wireless devices that are used to collect information at the point of delivery and collection would be the next logical step. At the present time these devices are usually built around ruggedised PDA style units that do a specific job and the applications that run on these customized devices are also custom built and don't carry other services such as email / timesheets / calendars etc. They are built around the standard client server implementation where the interface between the two is customized, or built from scratch to fit in with the corporate infrastructure needs.

At the present time such specialized data collection software elements that are required by the customer have to be specified and written into a contract to supply. The whole specification and procurement process would change for businesses if the same wireless enabled PDA/UE devices that are used by the public and purchased over the counter could be utilized in the role described above. The major hurdle to overcome this barrier is the ability of developers to produce software components that can operate on any number of wireless computing platforms and connect through to hosts that can reside on any number of infrastructure elements. Then the business purchaser with a specific need can select COTS hardware, OS, ORB and software objects that suit his needs. Component oriented software applications in this instance then become another set of pieces in the jigsaw that help complete the picture. To continue the analogy it is not really sensible for a customer to be expected to build up the majority of the components in his wireless 'field force enabling system' using jigsaw pieces and leave one piece out. The hardware, OS and protocol stack components take the development most of the way but then the customer has to have to pay an artist to paint in the final few pieces to complete the picture in the form of the proprietary software that runs on the platform. Having an ORB on these kind of devices would allow the labor intensive, time consuming and costly activity of implementing software for each of the fashionable UE and PDA platforms to be more widely spread across more licensed sales. They then have the ability to reach a wider audience and provide for much better cost models through re-usability and longevity of the applications.

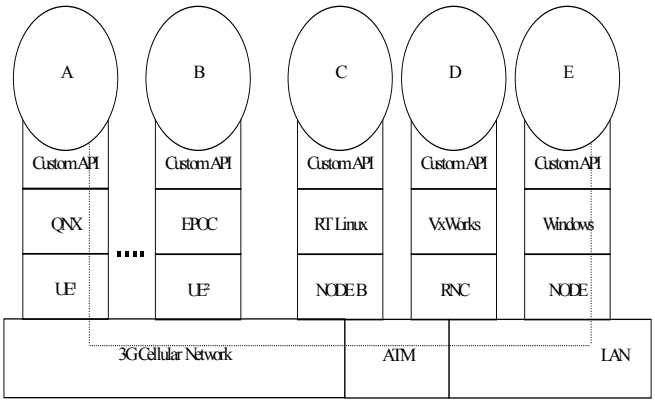

Figure 3-4 – Existing 3G Application Operating Environment

The wireless mobile and PDA market has multiple manufacturers competing for the hundreds of millions in hardware sales worldwide and on those devices there are the multitude of competing operating system suppliers waging war. The main client server application on offer right now across these UE/PDA hybrids is the basic service of email. Most PDA offerings allow dial in through the control of a co-located mobile phone and the opening up of a data channel to allow email synchronization with an ISP or the corporate email system, but this is where the client server commonalties end. Vendors then diverge from that point on and the support of client server applications across the multitude of platforms fragments.

As hardware production costs reduce and the sales volumes increase the price of the hardware unit to the purchaser can be reduced. In business (across all verticals) and high street sales the volumes of horizontal software sales warrant large development investment as the returns are there for the taking. However, within the business verticals or niche end user communities, client server systems that operate with some other entity in the enterprise are the norm. The effort required for extending such custom software developments to the wireless hand-held devices is always going to be prohibitive for any reasonable sized application. There is little benefit to suppliers of software that run on one platform, within one specific vertical for maybe only one specific customer, if all their sales revenue is offset against the development cost and like wise the clients who might want such services extending have to justify the expenditure in terms of man hours saved when compared to manual processing of the job.

If a CORBA environment is available across the range of UE's and/or wireless PDA's, then fabricating the client server custom applications that then run on the general purposes devices (in more durable shells) will not be the preserve of specialist software companies. The Middleware enabled applications that presently operate on nodes within the corporate infrastructure automatically and naturally extend to the wireless enabled field force users. Application and OS interfacing obviously has to be performed somewhere in the platform, OS, ORB, application layer, but by performing it at the ORB/OS interface the device manufacturer can gain from the increase in application availability and the application providers gain from the increase in compatible devices. The two entities become decoupled from each other and development of each can flow without the delay from the other, with the ORB vendor providing the lubrication.

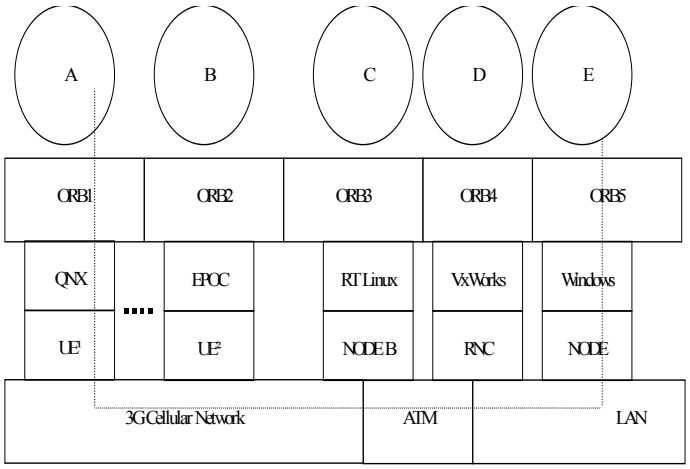

Figure 3 -5 – Potential for Middleware in the 3G Mobile Arena

Staying with the assumption that the predicted take up of 3G wireless devices is realized and CORBA ORB's are available on all these platforms, then the possibility of object 'A' in Figure 3 -5 above being able to communicate with object 'E' without having to have further protocol layers added, proprietary protocols developed or service providers hosting and charging for the privilege is plausible. What was termed the 'Personal ORB Framework' earlier in this paper is fleshed out here. One example use of the Personal ORB Framework might be for remote control of the users home appliances. If the node on the LAN that application 'E' ran on were an embedded device in part of a wireless home network, and that embedded device controlled the hosts central heating environment, then the UE node objects ('A'), could control the central heating node through the ORB using the published methods (through a home PC server acting as arbiter). Similarly the client object on the UE could interrogate a servant on the refrigerator node on the users Personal ORB Framework and report back what the contents were (or were not) present depending on the preferences set.

Although the examples given above could be implemented now using conventional technologies, the development of custom applications to support the control required is where the difficulties arise. All the manufacturers of embedded wireless devices that would reside in home appliances would need to have a corresponding control client developed for the multitude of platforms that the user might want to operate them on. By placing an ORB over the platforms, proprietary or standard interfaces could be operated without regard to

location or platform considerations of the client so long as the IDL description and client code is available for use.

This ability to control objects regardless of physical location and platform or physical communication medium can be easily extended to other areas. Other application areas of ORB based objects that reside on wireless embedded devices could be in the field of industrial plant control. If common CORBA services are adhered to and made available this greatly improves the situation, but not having this set of well defined services does not preclude the integration of such distributed components using a CORBA framework. In the above cases it is of course paramount that the controllable objects are adequately protected from outside attacks and a degree of security needs to be present. By making use of the CORBA security building blocks, human users or other objects in the distributed system can identify themselves and be denied or granted access to the objects methods they might want to control.

Having the ability to draw upon the existing CORBA security services now, opens a whole panorama of possibilities when CORBA frameworks are overlaid on embedded wireless devices. Extending the example of the CORBA based wireless distributed home network or the industrial plant control systems into the utility supply domain then water, gas and electricity metered supply points within the users premises could be fitted with low power wireless devices that periodically provided their users details, device status and supply reading information over a cellular network maintaining full end-to-end security. Again, possible right now on GSM technology, but implementation of the software and services for such applications would be costly for such a narrow scope. Having the ability to select components that are pre-built and fulfill a range of control applications that operate on an ORB eliminates the bespoke nature of the work and would bring down the cost of such custom applications. In that way the whole of the utility providers (nationally or internationally) could re-use components without having to each bear and pass on their costs to their own relatively customer base.

In theory this use of CORBA on wireless embedded computing devices is sound but the ORBs that give rise to theses sorts of services will need to be CORBA compliant. As discussed a fully specified ORB will be too weighty to support on an already application crowded hand-held embedded device, so judicious choices will need to be made. In effect an embedded CORBA ORB with the minimum set of CORBA services would need to be selectively built.

Evidence is at hand to suggest that these very types of tight, light, real time embedded target ORB's are becoming available and the use of CORBA technology in the infrastructure is impending.

Even if the CORBA framework does not extend to the wireless UE elements, network operators and managers still need real-time access to information about how the wireless network is performing at any given time. If ORBs and the CORBA objects are rolled out to the edges of the infrastructure then powerful analytical tools that help model trends and enable future decisions that impact service quality can be more easily integrated into transparent management reporting tools.

4. CONCLUSIONS

A prime example of a direct benefit that could be obtained by OEM's creating UMTS Node'B' and RNC equipment and applications would be in the development of the application parts to operate on a CORBA framework. The infrastructure development costs for 3G are very high and many companies are competing viciously in their roll out program of 3G services. Because the specifications are so complex many problems have been found along the way and software development programs have slipped as a result. Even if different hardware were available from the competitors on the basis of cost/quality/scalability/availability/etc. then so long as the software application parts had well defined interfaces to manage the hardware resources (through an ORB), it really would be a case of write once run anywhere. When well defined interfaces are provided by the likes of ETSI/3G/etc. then this lead should be taken. In a sense this arrangement is mirroring car production techniques where by different manufacturers collaborate on a chassis, share a parts bin but just assemble badge and market the final product differently. The ORB in this analogy would be the chassis and the CORBA services the engine, gearbox, drive train and wheels. Different combinations of these will give the car its ability to drive on the varied (operating system) terrain. The body parts and interior trim would be the objects that the user sees and interfaces with (feature rich or in some cases spartan).

This analogy then leads on to the problem of how the development of telecommunications software projects and processes are affected by this rapid application development. Although this is not a direct CORBA issue and is more a management problem, but it does raise the question of how this kind of issue is approached. As development programs are divided and spread amongst a number of collaborators the individual component developments then extend to different sites. To fulfill large and complex applications like the NBAP (Node'B' Application Part) then each may end up being managed in different ways on different platforms that are eventually destined to work together. It is felt that these types of project can only succeed when middleware is employed. By employing CORBA on this scale of development the once previously hidden interfaces are opened up through the IDL that can then be used as independent test points to ensure that acceptable software quality levels are met.

Partitioning the elements and distributing them over a number of sites has other benefits also. The resources that might need to be applied in fulfillment of some task or another could be assigned to a company in a developing country whose labour costs are lower and in supplying only component parts the IPR of the whole application is not compromised. Once the bulk of the objects required are pulled together then the application

can integrated, tested and supported by the prime contractor. So long as each object is specified to be implemented in a specific language for a specific ORB then any developer with CORBA and the identified language knowledge will be able to be assigned a role and will become reusable throughout the project. At the present time many telecommunications software engineers have highly transferable common language skills. To a certain degree portable operating system skills but many are restrained by their protocol skill sets. If the telecommunications sector embraced component oriented developments then the available work force would increase, become more flexible and costs would ultimately reduce.

On reflection, the concluding remarks with respect to employing CORBA in the mobile telecommunications sector the argument has to be: for application on the network edge, but is against the deployment within the infrastructure on practical grounds. Even though the many significant benefits already provided here could be derived if CORBA were adopted, the necessary momentum required to take the industry in this direction is not yet evident.

REFERENCES

[1] Folientitel, Kein . Distributed Systems – CORBA. Retrieved November 2008, from http://www.cse.cuhk.edu.hk/~xychen/Tutorial_PPT/DS/Distributed%20systems_CORBA.ppt

[2] Christopher, D., Kuhns, F., Levine, D. & Schmidt, D, Applying Adaptive Real-time Middleware to Address Grand Challenges of COTS-based Mission-Critical Real-Time Systems, Proceedings of the 1st International Workshop on Real-Time Mission-Critical Systems: Grand Challenge Problems, IEEE, Phoenix, Arizona . (1999).

[3] Hurwitz, Judith. Sorting Out Middleware. Retrieved November 2010, from http://www.dbmsmag.com/9801d04.html (1997)

[4] Vepstas, Linas. Linux DCE, CORBA and DCOM Guide. Retrieved November 2010, from http://linas.org/linux/corba.html (2001)

[5] Siegel, Jon, CORBA 3 Fundamentals and Programming (2nd Ed.). New York: John Wiley & Sons Inc. (2002)

[6] Burden, Peter , PC Networking. Retrieved November 2007, from http://www.scit.wlv.ac.uk/~jphb/comms/esppt/esppt1/sld045.htm

[7] Cullen, Drew, Sendo Junks MS SmartPhone, Joins Nokia Camp. Retrieved November 2010, from http://www.theregister.co.uk/content/59/27986.html (2002)

[8] Coulson, Geoff, What is Reflective Middleware. Retrieved November 2006, from http://dsonline.computer.org/middleware/RMarticle1.htm (2002).