Infrastructure Support for Adaptive Mobile Applications

Adrian John Friday

B.Sc. Hons. (London, 1991)

Computing Department,
Lancaster University,
England.

Submitted for the degree of Doctor of Philosophy,
September, 1996.

Abstract

Infrastructure Support for Adaptive Mobile Applications

Adrian John Friday

B.Sc. Hons. (London, 1991)

Computing Department,
Lancaster University,
England.
Submitted for the degree of Doctor of Philosophy,
September, 1996.

Recent growth in the number and quality of wireless network technologies has led to an increased interest in mobile computing. Furthermore, these technologies have now advanced sufficiently to allow “advanced applications” to be engineered. Applications such as these are characterised by complex patterns of distribution and interaction, support for collaboration and multimedia data, and are typically required to operate over heterogeneous networks and end-systems. Given these operating requirements, it is the author’s contention that advanced applications must *adapt* their behaviour in response to changes in their environment in order to operate effectively. Such applications are termed *adaptive applications*.

This thesis investigates the support required by advanced applications to facilitate operation in heterogeneous networked environments. A set of generic techniques are presented that enable existing distributed systems platforms to provide support for adaptive applications. These techniques are based on the provision of a QoS framework and a supporting infrastructure comprising a new remote procedure call package and supporting services. The QoS framework centres on the ability to establish explicit bindings between objects. Explicit bindings enable application requirements to be specified and provide a handle through which they can exert control and, more significantly, be informed of violations in the requested QoS. These QoS violations enable the applications to discover changes in their underlying environment and offer them the opportunity to adapt.

The proposed architecture is validated through an implementation of the framework based on an existing distributed systems platform. The resulting architecture is used to underpin a novel collaborative mobile application aimed at supporting field workers within the utilities industry. The application in turn is used as a measure to gauge the effectiveness of the support provided by the platform. In addition, the design, implementation and evaluation of the application is used throughout the thesis to illustrate various aspects of platform support.

Declaration

This thesis has been written by myself and the work reported herein is my own. The research has been carried out within the MOST project, a collaborative project between Lancaster University and EA Technology Limited, and jointly funded by the EPSRC/DTI (grant number GR/H87704).

The overall approach towards adaptation and the supporting QoS framework, based on the separation of concerns, benefited from discussions with colleagues from Lancaster involved with the MOST project. The design and implementation of the computational and engineering support for the framework, which forms the new distributed systems platform (sections 6.3, 6.4 and 6.5), were developed by myself. The collaborative mobile application presented in chapter 5 was developed jointly by the MOST project team. However, the audio conferencing facilities, lightning warning service and E-mail support within the application (sections 5.3.3, 5.3.4.2 and 5.3.4.4) were developed by myself. The Base Services, which support the audio conferencing module, benefit conceptually from earlier work at Lancaster on RM-ODP compatible services for handling continuous media.

The work reported in this thesis has not been previously submitted for a degree, in this or any other form.

Acknowledgements

Firstly, I would like to offer my thanks to EA Technology for providing sponsorship during the first three years of my Ph.D. EA proved to be an invaluable source of background information and experience, which shaped the research presented in this thesis. I would especially like to acknowledge George Worship, primarily for his friendship, but also for answering my numerous questions concerning the power distribution industry in the U.K.

I would like to thank my friends in and around Lancaster for helping to create an environment in which to live and work that, as anyone who has lived here will tell you, most never wish to leave. In particular, special thanks are due to Doctor Nigel Davies, sounding board, mentor and friend, without whom this Ph.D would never have seen completion.

I would also like to express my thanks to my supervisor, Professor Gordon Blair, who has not only taken time out of his sabbatical, but precious time away from his wife and burgeoning family to assist in the preparation of this document. This thesis would not have taken shape without his incisive and constructive criticism.

Finally, I wish to formally acknowledge the not inconsiderable role played by my family over the years. Although my parents may never fully understand why I have chosen the career path I am pursuing, in preference to more lucrative opportunities in London, their support and encouragement has never wavered. Above all, I would like to extend my heartfelt thanks to T.J., my girlfriend, who probably did not fully realise quite what a sharing experience my Ph.D was to become. Without her undying encouragement and affection to buoy my spirits during the trying times, I most certainly would not have been able to stay the course.

September, 1996.

To those who shared the burden, yet offered only boundless faith and selfless encouragement.

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Overview

The increasing development of portable computers and the widespread deployment of wireless networking technologies has led to the emergence of mobile computing as a major research area. In parallel, heterogeneous networks and end-systems have encouraged the development of open distributed processing standards and compliant platforms. The operation of these distributed systems platforms within a mobile context raises many important issues. This thesis examines the question of providing distributed system support for mobile applications in open heterogeneous systems.

This chapter discusses the emergence of distributed systems platforms and of the open distributed processing standardisation. The chapter goes on to discuss the primary implications of operation within a mobile environment, particularly from the viewpoint of advanced applications. A mechanism for providing support for these applications within a distributed systems platform is proposed. Lastly, the aims and objectives of the thesis are enumerated.

1.2 Distributed Systems

To gain a perspective on the problems associated with using conventional open distributed processing platforms in mobile environments, it is first necessary to review their development and the key benefits they provide to application programmers.

1.2.1 Emergence of Distributed Systems

The integration of computer and communication technologies has revolutionised modern computer systems. Local computing resources are interconnected via easily extensible networks to form highly cost effective and scalable *computer networks*.

Traditionally computer systems were a centralised resource with access provided by terminals connected via serial links (e.g. RS232). Computers such as these needed to be very powerful to provide enough CPU time for user processes and still give satisfactory response times for interactive users. However, these systems do not scale well: as the number of users increases there is often a small degree of expansion possible but ultimately the system must be replaced at a considerable price. In contrast, a network can be extended by adding supplementary computers on demand at comparatively little cost.

In addition, computer networks enable resource sharing, improve resource availability and, through redundancy of hardware and software, can increase fault tolerance. Moreover, networks enable systems to be developed which more accurately reflect the naturally distributed nature of the workplace. Building systems which are capable of operating within networked environments such as these can be simplified by additional software.

Layers of software can be added to a network of computers which abstract away from details about system components and their interconnections, enabling the network to be viewed as a single *distributed system*.

> "*... the distinction between a network and a distributed system lies with the software (especially the operating system), rather than with the hardware.*" *[Tanenbaum,88]*

A number of the more important ways in which details can be hidden or *forms of transparency* provided are outlined below (in the following breakdown a process, object or system resource will be described as an item) :-

Location — If the location of an item cannot be determined, the system provides location transparency.

Access — If the procedure for accessing an item is identical, regardless of physical location, the system provides access transparency.

Migration — If an item can be moved to another location without interrupting the service it provides then the system provides migration transparency. Migration transparency is most often applied to processes and objects, typically physical resources do not move. However, the ability to

access a resource can be migrated (e.g. ports in CHORUS [Herrmann,88]).

Concurrency — If access to an item by multiple parties is managed transparently (by for example, enforcing the serialisation of each party's actions) then the system provides concurrency transparency.

Replication — If multiple items can be kept within a system to improve availability and fault tolerance but still appear to clients as a single item then the system provides replication transparency. Replication transparency is synonymous with group transparency.

Failure — If the partial failure of an item (set of replicated items) is invisible then the system provides failure transparency.

The forms of transparency outlined above, which represent a sample of the many forms available, can be combined together to form powerful distributed systems. If all of the forms of transparency were to be implemented within a single system (assuming it is possible to do so) then such a system would appear to be a virtual uniprocessor with no apparent boundaries between individual computers. However, the complete transparency offered by such a system is, in many cases, undesirable. For example, consider a surveillance system. Each video camera within the system would offer a video service of the required type. The selection of the appropriate service would depend, firstly, on knowing the location of the service which offered the particular view that the client was interested in and, secondly, on being able to access that service. Thus in real distributed systems what is often required is *selective transparency*: the ability to choose which forms of transparency are provided by the system to any given client.

The heterogeneous nature of wide scale integrated networks however make engineering forms of transparency very difficult, if not impossible. For instance, consider the implementation of migration transparency. A number of issues are raised, including how to maintain versions of executables for each architecture, whether to combine these versions to form a single large executable or maintain them separately, and when to compile each version. In addition, when one of these versions is running and needs to be migrated, the state of the process will need to be transferred despite the potentially different data representations on each architecture. Clearly there is a considerable overhead in maintaining these libraries of executables and providing translators for passing execution state between versions. As an alternative, one could employ a more general mechanism whereby computers executed a universal interpreted language with an interchangeable data set. However, despite the ease of

migration, the performance overhead would be substantial. It is unsurprising therefore, that transparencies such as these are most often employed on systems spanning common architectures. Nevertheless, implementations of more general distributed systems containing many of the easier realised transparencies are available. Many of these platforms conform to *open distributed processing* standards whose goal is to ensure that the platforms can interwork to provide these transparencies in a heterogeneous environment. The main activities toward standardisation of these platforms are outlined in the following section.

1.2.2 Open Distributed Processing (ODP)

There are three main efforts toward developing standards governing open distributed systems platforms (an outline of each is presented below). These emerging standards specify distributed programming paradigms which support a number of the transparencies previously discussed. The aim of each standardisation activity is to gain acceptance for their framework and, in some cases, implement viable platforms to demonstrate the concepts they embody. A thorough discussion of the aims and objectives of the following activities and their associated standards can be found in [Adcock,94].

ISO Open Distributed Processing (ODP)

The ISO define a Reference Model for Open Distributed Processing (RM-ODP) [ISO,92] (akin to the widely accepted ISO/RM communications network model). The model defines a set of terms and concepts for describing distributed processing, a generalised distributed processing model according to the set and a general framework for contrasting relevant open standards. The Advanced Networked Systems Architecture (ANSA) project [APM,89] has been a major contributor to the RM-ODP.

Open Software Foundation (OSF)†

The OSF are an industrially sponsored organisation whose aim is to provide openness from integration of selected software technologies. Their standards are intended to gain support by guaranteeing stability and continuity to potential customers. Support for distributed applications in an open environment is provided by a platform known as the Distributed Computing Environment (DCE [OSF,91]). DCE is

† Following a merger with the X/Open organisation at the start of 1996, OSF and X/Open are now known collectively as “The Open Group”.

comprised of standardised components including threads, RPC, directory, time and authentication services.

Object Management Group (OMG)

The aim of the OMG is to integrate existing standards to provide a comprehensive distributed open systems framework. To achieve this goal the group are explicitly applying object oriented technology. Specified services from the architecture include a naming service, an event service (for inter-object communication), life cycle services (for object management), an association service and a persistence service. OMG offer an architecture known as the common object request broker (CORBA [OMG,91]). CORBA offers functionality similar to DCE, providing inter-operability between machines and seamlessly interconnecting multiple object systems. CORBA is entirely defined in terms of interfaces to objects and is consequently not limited to any particular implementation.

1.3 Advanced Applications

The widespread use and interconnection of networks allows us to communicate with other individuals in institutions world-wide. Public enthusiasm for the Internet has spawned a number of Internet service providers. The revenue which is generated will inevitably lead to improved network infrastructure to cope with the increased demand.

With the steady improvement in the speed and quality of networks, the ability to spontaneously transmit large quantities of data between multiple points has led to considerable research into *multimedia* information distribution (for example, multimedia conferencing [Lantz,86] and distance learning [Mason,89]). The transmission of multimedia data places stringent requirements on a network. For example, a typical stream of video from a camera may be of unbounded length and impose timeliness constraints on packet delivery to maintain continuity. Requirements such as these are referred to as quality of service (QoS) requirements.

The need to transmit multimedia information has fuelled a long running debate over the kind of services that a network should provide. There are a number of approaches, which essentially range between two extremes: *best effort* and *guaranteed*. The best effort approach [Danthine,92] is to manage with the currently available resources and introduce mechanisms to cope with the wide variation in network load that will inevitably arise. The opposing approach is that the network

service provider should guarantee that the necessary resources will be reserved within the network for each data stream based on application specified QoS.

Other proposed solutions include statistical [Campbell,93] and, latterly, compulsory [ISO,95b] (where resources are reserved according to the average demands of the application), predictive [Käppner,94] (where the end-system uses only the resources that it predicts will be available based on observed resources in the past) and adaptive [Käppner,94] (where the end-system attempts to meet the specified QoS by modifying the source of the data stream in response to feedback about the network).

In parallel with these developments in the transmission of multimedia information, the ability to exchange information within a network has fostered a considerable quantity of research into supporting group working practices; more specifically entitled computer supported cooperative work (CSCW). This research covers a wide range of topics from assisting groups to solve a particular task, to helping people overcome geographical separation. A good discussion and categorisation of CSCW can be found in [Rodden,92]. The collaborative exchange of multimedia information requires efficient and scalable multicast and broadcast protocols to minimise the consumption of network bandwidth (for example, the Scalable Reliable Multicast (SRM) [Floyd,95] framework which underpins the “wb” shared white-board tool [Jacobson,92a]). Researchers have noted that, in a given multiparty interaction, each participant is likely to have their own particular information requirements based on their end-system capabilities, user requirements and, significantly, the capabilities of the networks they are connected to. Based on this observation, filtering techniques [Yeadon,94] can be applied which reduce the overall bandwidth demands of continuous media applications by hierarchically reducing the content of the streams of information downstream from the information source in response to each participant’s demands.

Applications characterised by complex patterns of interaction, incorporating aspects of cooperation and use of multimedia, particularly over heterogeneous networks and end-systems can be called *advanced applications*. Such applications, are typified by group interaction and control, peer-to-peer information exchange and integration of multimedia data types.

The highly complex nature of advanced applications makes them particularly difficult to engineer. Research has shown that with the aid of a suitable distributed systems platform the development of challenging applications can be simplified. For example, the ISIS platform [Birman,90] is able to greatly simplify the development of group based applications. Furthermore, the DASH distributed systems platform

[Anderson,90] provides mechanisms to enable the construction of continuous media based applications. It is the author's opinion that the complexities of building future advanced applications in heterogeneous environments will require the mediation of a distributed systems platform.

1.4 Impact of Mobility

The availability of wide-area wireless networking technologies has led to the development of a number of data services. Pocket sized cellular modems have allowed standard analogue cell-phones to be used as data service carriers. Typically these links are of poor quality with high bit error rates. However, techniques such as MNP5 make data rates of around 2.4Kbps possible.

More recently, the deployment of digital cellular services such as GSM has opened up the possibility of reliable communication at up to 9.6Kbps. In wide-area wireless networking terms these links are seen as reliable. However, the bit error rates are still two to three orders of magnitude more frequent than present in fixed networks. In addition to the voice traffic specifications, the GSM standard incorporates provision for data services. The data service specifications have been followed to a greater or lesser extent by different suppliers. To date however, no supplier has implemented group data support. The CDPD service operated in the United States offers an alternative approach to data service provision. The CDPD service is retrofitted to existing analogue cellular systems to offer connectionless store and forward packet services, well suited to low bandwidth IP use. These wide-area data services offer reasonable, if costly, connections to fixed networked resources.

The greatest potential benefit for wireless networked systems can be derived through an integration of wireless technologies to provide seamless connectivity as a user roams. For example, a mobile user may be able to utilise a wireless local-area network within their office environment and, on leaving the building, seamlessly switch to wide-area wireless service provision. However, the consequences of user mobility has far reaching implications, particularly in terms of the assumptions made by conventional systems. A number of the more common assumptions affected by mobility are presented below :-

Variable connectivity

In an integrated networked environment the QoS offered at any instant by the network is variable. In the above example the user moves from a packet-oriented local-area network offering approximately 2Mbps to a wide-area service offering at most 9.6Kbps. In addition, the wide-area

service may introduce other factors such as dial-up delays and increased bit error rates.

Resource availability

Applications are typically constructed with target environments in mind. For example, file systems can consider that their servers are readily available and that network partitioning is a rarity. In systems comprising mobile components, the availability of a given resource is far from certain. Typical wide-area wireless systems offer, at best, intermittent connectivity and often experience drop-outs and communication blackspots.

Resource location

The location of a resource is often taken for granted within networked environments. For example, the Internet suite of protocols identify each host in the network by a four byte address which hierarchically denotes that host's location within the network. Furthermore, the identifier forms the basis for all routing of information to and from that host. The changing location of a mobile host has profound implications for the use of services on and by that host, particularly in an environment where the host may be available via a range of independent connecting technologies. For instance, a migrating host using a particular service may wish to handover to a more appropriate but equivalent service at the new location.

Cost

To many users the cost of using a networked resource can be considered free. Furthermore, most applications pay precious little attention to the amount of network resources they require on this basis. In a system which integrates premium rate public services, a user can incur considerable costs for using remote resources. Furthermore, depending on the service in use, the cost and basis for charging will change. For example, a GSM network will charge for connection establishment and per unit time that the connection is held open. A CDPD network will charge per unit of information that is sent. Clearly users will require applications to access resources in as cost efficient a manner as possible.

Power consumption

Within conventional networked environments power consumption is barely a factor. To a mobile user however, the battery life of their

system is of primary concern. Portable end-system designers have recognised this issue and incorporate clever power saving techniques into the system hardware to help maximise battery life (e.g. aggressive disk spin down policies and low power CPU doze modes). However, the integration of communications technologies presents a further drain on battery resources: each item of data exchanged represents a reduction in battery life.

The key characteristic that mobility in a heterogeneous networked environment presents is *change*: change in service location, change in the QoS offered by the network and change in the availability and penalties associated with the access of services. As Katz highlighted:

"*Mobility requires adaptability. By this we mean that systems must be location- and situation-aware, and must take advantage of this information to dynamically configure themselves in a distributed fashion.*" [Katz,94].

The author believes that it is only through a process of *adaptation* [Davies,94]: providing applications with managed information about changes in their supporting infrastructure that they can hope to operate efficiently in a highly dynamic environment. Moreover, such applications may in turn provide feedback, enabling user expectations of the system to change and thus implicitly change their mode of working to make better use of the available resources.

Given, as previously stated, that in order to develop advanced applications a suitable distributed systems platform is needed, combined with the necessity for the provision of an architecture which facilitates application adaptation, it stands to reason that the distributed systems platform must provide this supporting architecture. However, the provision of such an architecture within distributed systems platforms is non-trivial: there is clearly a conflict of interest between adaptation which requires the lower-level information to be made available and the forms of transparency traditionally provided by platforms which are intended to hide lower-level details in abstraction. Moreover, the provision of these fundamental forms of transparency are impacted upon by operation within a mobile setting.

For example, consider location transparency. A client which has access to two identical services, one via a fixed network and one mobile, may pick either service. However, if the mobile service is chosen, the variable bandwidth and potential lack of availability would have a significant impact on system performance. Alternatively, it may be essential to obtain a mobile service, where it equates to a particular user for example. In addition, considering location transparency combined with group or

replication transparency, the performance of a system would be heavily impacted by the inclusion of mobile group members.

The aims and objectives of the remainder of the thesis, which addresses a number of the issues of providing mobility support within distributed systems platforms set out above, are discussed in the following section.

1.5 Aims

This thesis investigates the issues of providing infrastructure support to advanced applications in heterogeneous networked environments. In particular, the thesis focuses on the role of adaptation within distributed systems platforms: the notion of being able to partially break transparency, enabling applications to specify their requirements and be provided with managed information in order that they may adapt to changes in their environment. More specifically, the research has the following goals :-

- To investigate the state of the art of mobile computing and provide an evaluation of existing research and supporting wireless technologies.
- To study the support requirements of mobile applications which would enable improved operation within heterogeneous networked environments.
- To develop a set of generic techniques which enable current distributed systems platforms to support adaptive mobile applications.
- To implement a specific adaptation enabling technology within a distributed systems platform: i.e., an enhanced remote procedure call package and associated binding mechanism known collectively as QEX.
- To evaluate the new technique through the design, implementation and evaluation of a collaborative mobile application[†], underpinned by the enhanced platform.

It should be noted that while the research is intended to enable applications to deal with a number of the implications of host mobility (outlined in section 1.4), the issues relating to power consumption have not been addressed in this thesis.

[†] The application was developed as part of a joint project involving Lancaster University and an industrial end user organisation (the MOST project). See the declaration for a precise breakdown of the research conducted by the author.

1.6 Thesis Outline

The remainder of this thesis is structured as follows. Chapter 2 contains a discussion of current hardware and communication services available to support mobility. In addition, the chapter explains how these technologies are being inter-linked to offer continuous connectivity.

Chapter 3 analyses a wide range of current mobile computing research including system and application level software. The chapter argues that the majority of this work does not provide sufficient support for the development of advanced applications within heterogeneous mobile environments.

Chapter 4 describes the rationale for the advanced application, developed as part of the MOST project, which is used to help evaluate the supporting platform. The requirements which the application addresses and the scenario which led to the requirements are presented.

Chapter 5 describes the prototype application that was realised to address the requirements presented in chapter 4. The design and structure of the application are described in detail, with particular emphasis on the features which improve its suitability for operation in heterogeneous networked environments and, more specifically, enable it to adapt. The mobility support that the platform is required to provide to the application is highlighted.

The components of the support platform are examined in chapter 6. The model upon which the platform is based is explained and a range of enhancements presented that are designed to support application adaptation. For clarity, the enhancements are compared and contrasted with the more familiar RM-ODP model. Each component is then described in detail, particularly in terms of how it adapts to changes in the environment and provides feedback to other layers. The chapter concludes with a brief discussion of a range of new lower-level support services that are required by the platform for operation in a mobile environment and that are not currently provided by existing transport protocols and device drivers.

Chapter 7 evaluates the platform in the light of how effectively it was found to support the mobile application. The new computational model and engineering implementation are evaluated, particularly in terms of suitability and convenience of use. Performance figures are presented for the communications oriented components of the platform. Support for some of the more important elements of the application such as decentralised state, group management and managing multimedia information are assessed.

Finally, chapter 8 summarises the main points of the thesis. The most important results are highlighted. The chapter concludes with pointers to areas where future work is required.

Chapter 2

Enabling Technologies for Mobile Computing

Recent years have witnessed the emergence of the field of mobile computing as a significant research area. Pioneering work such as the Columbia/IBM Student Electronic Notebook (SEN) [Duchamp,91], the Berkeley InfoPad [Keeton,93] and Xerox's Ubiquitous Computing project [Adams,93] have demonstrated some of the potential benefits of using local-area wireless connectivity. More recently, the availability of cheap wireless local-area networking products has led to increased deployment in research and industry. In addition, the availability of an extensive range of high coverage, low bandwidth communications services is fostering increased integration; both with end-systems and other communications technologies.

This chapter aims to provide an overview of the currently available portable end-system and wireless communication technologies. In addition, the chapter hypothesises that these local and wide-area technologies can be integrated to offer continuous variable bandwidth connectivity.

2.1 End-system Technologies

End-system technologies can be loosely grouped into two independent development tracks: *portable computers* (PCs) and *personal digital assistants* (PDAs). These two classes represent the most common platforms for mobile computing research and development. This section charts the development of the two technologies and outlines their suitability as potential mobile application development platforms.

2.1.1 Portable Computers

Portable computers are the result of a continuing trend in miniaturising desktop personal computer hardware for portable use. Portables offer the same set of features as their desktop counterparts in one fully integrated package, typically comprising a flat screen monitor (most commonly flip up), a keyboard, mouse (or replacement), floppy and hard disks and, increasingly often, integrated speakers and microphones. The primary limitation to further reduction in size is the physical display area of the screen.

At the time of writing, portable computers are available with Intel Pentium 200MHz and PowerPC 166MHz CPUs. These portables represent at least a factor of four increase in raw computing performance over the machines offered just two years earlier. LCD technology can now offer colour SVGA resolution of 1024x768x8. Internal peripherals such as removable $2^1/_2$" gigabyte hard disks and integral CD-ROM drives are commonplace. In short, portables offer the research community extremely powerful machines often, in performance terms, outstripping previous generation low-end workstations. Significantly, these portables are now sufficiently powerful to run research quality operating systems such as UNIX and WindowsNT.

Generally though, portable computers belie their heritage: the need to remain compatible with their desktop counterparts forces a number of architectural constraints upon them. Most importantly, the tie to a specific chipset (Intel or Motorola) means that modern low power processors cannot be used. Fast but power hungry processors coupled with high definition colour LCD screens and peripherals such as radio modems and CD-ROM drives can bring battery life down to inconveniently short durations. The need to support the same windowed environments, specifically the mouse, has led to the integration of a number of ingenious solutions such as the trackball, the touch sensitive pad and even a tiny pressure sensitive rubber nodule located between the keys of the keyboard (similar to a small joystick). These input devices are neither as convenient nor as intuitive as a pen based interface, particularly if used on the move, but are again retained for backward compatibility reasons. One exception worthy of note is the GridPad which retro-fits a pen interface to a Microsoft Windows derivative known as PenWindows. The GridPad also features a novel flip-over LCD screen which allows its use as an entirely pen driven electronic pad or as a more conventional “book” style portable with an integral keyboard.

The reduction of personal computer hardware to the approximate size of a ream of A4 paper has placed serious restrictions on the expandability of the PC bus architecture. In response, the solution has been to integrate more hardware onto the

motherboard as standard, for example sound sampling hardware, serial and parallel interfaces. Although a number of notebook machines have been designed with a single full size PC expansion slot, a more convenient form of expansion was required. Since June 1989 the *Personal Computer Memory Card International Association* (an association of software, computer and peripheral manufacturers, more commonly known as PCMCIA), have been developing a non-proprietary standard for credit card sized memory modules (which now encompass a variety of hardware including modems and LAN interface cards). Cards specified under the PCMCIA standard use a 68 pin interface and are 86.6 millimetres long by 54 millimetres wide with thicknesses of 3.3, 5 or 10.5 millimetres (types I, II and III respectively). Notebook PCs typically offer a single type III slot which doubles as two type II slots. Some manufacturers support a type IV slot larger than type III (Toshiba is one example), however this is not an official PCMCIA designation. Recently, interface cards which comply to the PCMCIA standard have begun to be referred to as PC card compliant.

Portable computers have been developed that are physically smaller than the typical A4 notebook sized PC. These portables are often referred to as sub-notebook or palmtop PCs. One example is the 100LX and 200LX range of palmtops from Hewlett Packard. Although capable of running the same software as other portable computers (with a reduced capacity display), their small physical size allows them to compete in the same market as personal digital assistants (see below).

A further derivative of the portable computer is the “electronic notebook”. An electronic notebook is a fully bespoke system intended to replace conventional paper notebooks. For example, the SEN project at Columbia University set out to develop an electronic notebook which would be able to download course notes and exercises automatically on entering a classroom and allow students to make notes and complete the exercises using a paper-like pen driven interface. These electronic notebooks should not be confused with the term notebook which is colloquially applied to conventional portable computers.

2.1.2 Personal Digital Assistants (PDAs)

Personal digital assistants are computers whose development is driven by completely different objectives to those of PCs. PDA design is based on a similar premise to the calculator: the provision of a set of useful tools (typically organisational) with no requirement to support any specific hardware architecture. New tools can be written at a variety of levels of hardware abstraction such as assembly language or C. In addition, most PDAs support a portable high level scripting language (one example is NewtonScript on the Apple Newton [Bey,93]).

The goal of the PDA designer is to produce a computer which can be comfortably held in the palm of one hand with a “natural” user interface, typically touch sensitive or stylus driven. PDAs provide integrated environments with tools such as a diary, notebook, address book, calculator and usually a small database.

For a PDA to become accepted as a viable alternative to existing personal organisation tools, it must be equally convenient and reliable. Consequently, PDAs must be very portable, rugged and exhibit long battery life. To address this last concern, PDAs use sophisticated power saving architectures combined with low power grey scale reflective LCD screens (which are lighter in terms of weight and power consumption than the colour LCD screens popular with portables).

Two common PDAs, the Apple Newton and Casio Zoomer, both use handwriting recognition as their user interface. Both products aim to translate printed and cursive writing into ASCII text, typically recognising up to 70% of user input (this is largely dependent on how much time is spent training the recognition software and how many of the words are stored in the dictionary). Handwriting recognition is a very complex problem and, although the ability of these PDAs improve with each generation, they are still are a long way from being natural to use. However, these PDAs excel at data entry applications where fixed format forms can be used that require simpler responses such as ticks and menu selections.

Some PDAs avoid the issue of handwriting recognition altogether. The Sony MagicLink and Sharp Zaurus are typical examples. Although employing a similar set of integrated tools to those previous mentioned, the pen interface is used to provide a point and click style of operation similar to a mouse or trackball. More sophisticated data entry is provided by an on-screen keyboard tool.

The Psion series 3 organisers use neither a touch screen nor a pen interface. Instead, a miniaturised QWERTY keyboard is provided. Rather than attempting to provide a paper-like interface, this approach does at least provide a familiar one. In addition to the conventional set of tools, the keyboard allows the series 3 to offer a powerful document editor (a tool which is not yet practical using handwriting recognition).

PDAs do have their limitations; their small physical size and requirement for long battery life constrains both processor power and expansion possibilities. PDAs typically contain a serial port and a single type II PC card slot. To help overcome the connectivity limitations most PDAs have a built-in infra-red (IR) transceiver which can *beam* information to other PDAs or fixed resources over a short distance (typically no more than a metre). This connectivity allows PDAs to provide

communication tools such as E-mail and Fax facilities and, more recently, world wide web browsers.

Some manufacturers have begun to integrate wide-area wireless communications into their hardware. Two examples, both from Motorola, are the Envoy and Marco *wireless communicator* products [Motorola,95]. In addition to a 38.4Kbps IR transceiver, the Marco incorporates a 4.8Kbps packet radio modem with 9.6Kbps fax capability. The Envoy includes a wireless packet data modem which operates using one of two protocols, MDC4800 giving 4.8Kbps or RDLAP at 19.2Kbps.

2.1.3 Analysis

Both portable PCs and PDAs provide facilities for mobile computing, although the two technologies lend themselves to different applications. The small monochrome displays and limited memory of PDAs make writing complex applications very difficult but long battery life and unobtrusive size make genuine use on the move possible. Portables offer remarkable processing power and local storage but the consequent short battery life places severe constraints on their use away from a fixed power source. All but the most expensive colour LCD technology is unusable in direct sunlight and in most cases the sheer size and weight make one handed operation unpleasant. Portables are best suited to applications where the user is mobile but does not need to use the machine in transit.

Both technologies utilise the highly pervasive PC card standard interface for expansion. The acceptance of this standard has prompted a large number of manufacturers to develop conformant versions of their products. Of particular interest in terms of mobile computing are PC card conformant wireless communications products. The remainder of this chapter considers a broad range of communications technologies and, where applicable, highlights specific products available on PC card format.

2.2 Communication Technologies

The previous section has considered the characteristics of portable end-systems. This section discusses wireless communication technologies which can be used to enable mobile applications to exchange information and make use of remote resources. The technologies have been grouped into two domains: wide-area and local-area. The former category pertains to systems which provide coverage over geographically large areas. In contrast, local-area technologies are designed primarily for in-building use, but typically offer far higher bandwidths. Standards and products relating to the technologies are considered where appropriate.

2.2.1 Wide-Area Communication Technologies

This section presents a summary of the more common wide-area wireless communications technologies. To aid understanding, the section commences with a discussion of some of the more important wireless networking concepts. The remainder of the section is concerned with discussing the technologies themselves. Each technology has been grouped into one of two domains: private services and subscriber services.

2.2.1.1 Basic Concepts

Wireless systems make use of a number of different techniques to provide wide-area coverage. One of the simplest approaches is to set up a single base station transceiver which serves a number of less powerful mobile transceivers. Each mobile is typically either hand-held or vehicle mounted for convenience. The coverage of the system is a function of visibility to the base station transmitter, interference from surrounding buildings and machinery, and the power of the base station and mobiles' transmitters.

A common extension to this approach is to divide the available communications bandwidth into a number of individual channels to increase the number of users which have access to a given portion of radio spectrum. This process, called *trunking*, requires a protocol to be established between the base station and the mobiles to determine which mobile is allocated to which channel and at what time. Each channel is made up of a combination of units of frequency, time or access pattern (frequency, time and code division multiplexing respectively). Typically, the channels are organised into a control channel and one or more data channels. The control channel carries instructions and requests to and from the mobiles to establish and clear down calls on each of the data channels. A non-trunked system does not layer a connection or call oriented structure over the network. Instead protocols can be used to control access to a shared channel "on-demand".

Another simple approach is the "open channel" scheme. Each mobile operates on the same frequency or channel and may listen to any transmission within range. Typically a social protocol is operated implicitly by the users of the system to dictate who may speak at any given time. The strongest transmitter (usually the nearest) wins. An extension of this approach is the "ad hoc" network. The ad hoc network has no formalised structure and makes use of the same "within range" peer-to-peer metaphor. Instead of voice traffic, data messages are routed across, and retransmitted by, each host to extend the coverage of the system. Ad hoc networks are only effective in sufficiently densely populated user communities. Moreover, each host

must be very cooperative and not mind using their battery resources to route other users' traffic for the good of the community.

The range of a single base station is limited by the transmitter power of the mobiles which is in turn a function of the required battery life of the unit (from elementary physics, to double the transmitter power requires a four fold increase in current from the battery). In addition, studies have shown that holding transmitters of any power close to the head increases the likelihood of brain tumours.

Each base station has a limited number of frequencies at its disposal. As mentioned previously, these frequencies may be multiplexed to increase the number of mobiles supported by the base station. However, for a given range of frequencies there is a finite limit to the number of the mobiles which can be supported by a given base station.

A common way of circumventing the power and capacity limitations is to operate multiple base stations (or cells) which are organised to cover adjoining geographical areas. The base stations are networked together which allows messages from one mobile to be routed via the base stations to a host in another cell thereby extending the effective coverage of the system. Cell sizes and locations can be chosen to reflect the density of the users. For example, high density areas would be covered by more numerous but smaller cells to increase the effective number of channels available.

Each neighbouring cell transmits on a different set of frequencies to allow for a margin of overlap between one cell and another without interference. The signal strength of the transmissions of the base station are used to work out when the transmission from one base station to a neighbouring one should be made. In addition, to set-up and clear down calls, the trunking protocol in a cellular system needs to manage the switching or *handover* of a mobile from one cell to the next.

Base stations typically power antennas mounted on terrestrial vantage points such as hilltops and the roofs of buildings. An alternative approach is to mount antennas on orbiting satellites. Generally these satellites are in a geostationary orbit, i.e. orbit at the same rate as the Earth and from the surface appear to maintain a fixed position in the sky. From a terrestrial perspective and, in contrast to a ground mounted antenna, the cells operated by non-geostationary orbiting satellites are seen to move with respect to the users (the velocity of the satellite is orders of magnitude faster than that of the mobiles). In addition to conventional power and interference concerns, satellite systems are affected by the incident angle between the Earth's surface and the satellites orbit, and also by atmospheric conditions.

2.2.1.2 Private Services

The following sections discuss the most commonly available private wide-area communications services. That is, the users of the system purchase all of the transceiver components including the mobiles and base stations. Once the system is installed the user is licensed to use the relevant frequencies of radio spectrum and it is their responsibility to operate and maintain the equipment thereafter. Privately owned radio services are most commonly referred to as *private mobile radio* or PMR systems. PMR systems can be separated into two distinct technologies: analogue and, the more recent, digital services. The following sections describe both types, beginning with analogue services and their associated standards.

Analogue Private Mobile Radio Technologies

Analogue trunked PMR systems generally conform to either MPT 1327/43 or PAA 2424 data signalling standards. These standards, defined in the U.K. and France respectively, are very similar, differing only in highly technical concerns. For the level of detail required in this chapter both standards will be considered equivalent and all discussion will focus on the U.K. standard (for a more technical analysis see [Wong,95]).

A MPT 1327 PMR system consists of a control channel and one or more data channels. The control channel is used to establish and clear down calls on the data channels. Additionally, Multi-site PMRs require the control channel to perform handover. A particular feature of the standard is the ability to send additional data on the control channel without establishing a call. The additional data may be either status codes (0 to 31) or free format data up to four concatenated codewords in length (25 ASCII characters). Later additions to the standard permit extended data messaging of up to 100 ASCII characters. The short message and extended message services are now referred to as single segment transactions (SST) and multiple segment transactions (MST) respectively.

The standard data rate supported is 1.2Kbps, although the MPT 1343 (superset of MPT 1327 applying particularly to trunked systems) standard allows nonprescribed data at rates up to 4.8Kbps.

Non-trunked analogue PMR standardisation has followed MPT 1317 (predecessor to MPT 1327) and PAA 1382 standardisation to form the BIIS 1200 (binary interchange of information and signalling [ETSI,93]) standard. The ETSI (European Telecommunications Standards Institute) approved BIIS standard supports reliable transfer in point-to-point and multidrop modes and an unreliable non-acknowledged

service. The data protocol is based on HDLC (ISO 4335 [ISO,87]) operating at 1.2Kbps.

Further standardisation has led to the drafting of an interface conformant to RS 232 and CCITT recommendation V.28 known as MAP 27 (Mobile Access Protocol for MPT 1327 [UADG,93]). MAP 27 operates nominally at 9.6Kbps, although 1.2Kbps is still the preferred option.

Digital Private Mobile Radio Technologies

In 1988 work the ETSI started work on the trans-european trunked radio standard (TETRA). Manufacturers were requested to make formal proposals to the STC-RES 06 subtechnical committee for a system which must offer both improved performance and more efficient use of available spectrum than existing protocols.

The TETRA standard, as it emerged, supports up to four channels in 25KHz of bandwidth (with an option that in the future it must be possible to support two channels per 12.5KHz of bandwidth). TETRA operates a TDMA structure based around a unit time slot of 14ms (chosen to be compatible with existing voice coding technology). Although this moderately long slot time could cause long end-to-end speech delays, TETRA is oriented toward simplex working and hence this delay is not a consideration. An interesting feature of TETRA is that inter-cell handover is initiated by the client equipment: the client can listen for stronger base stations and inform its current server to handover its connection to the new server; alternatively the mobile can manage the whole process by contacting the server itself (in this latter case a small discontinuity will be detected). While this approach leads to more complex clients, the technique is inherently more scalable than a centrally managed architecture and has potential for more flexible cell layout patterns.

The protocol has been designed to support voice, voice plus data or data only services. As with voice traffic, these other services are connection-oriented. One of the requirements of the TETRA standard was to have rapid connection establishment: the protocol can make use of successive free time slots in a lightly loaded system for control traffic bringing call set-up times under 100ms. In voice plus data mode, voice timeslots which contain no speech are “stolen” for data transfer (leading to high jitter and variable data rates). The protocol also supports a packet data optimised (PDO) mode in which all traffic time slots are monopolised in one bulk transfer. In PDO mode corrected data rates of 19.2Kbps are possible, although implementation of this mode is currently left at the manufacturers discretion.

Related Products

There are numerous companies selling PMR services, one of the more successful, particular in the U.S., is ARDIS. Initially an amalgamation of two PMRs owned by IBM and Motorola, ARDIS was designed to give links to mobile staff while inside the customer's building. ARDIS now has 1,300 base stations covering 80% of the population of the U.S. and approximately 90% of the prime business locations. ARDIS is now available as a public service, although focusing primarily on vertical applications for business subscribers. The service is being upgraded from 4.8Kbps to 19.2Kbps in the U.S. and 9.6Kbps in Europe (due to channel bandwidth allocation). IBM and Motorola both make radio modems for the ARDIS service, implemented on type II PC cards.

2.2.1.3 Subscriber Services

Subscriber services, in contrast to private mobile radio systems, are wholly owned by companies which specialise in offering communications services to others. These communications services may include the leasing of mobile terminal equipment and a multiplicity of charging and subscription arrangements for the accounting of user network traffic. This section describes two forms of subscriber service: proprietary data services and mobile cellular telephony.

Proprietary data services, generally speaking, have evolved to serve particular niche markets. This section describes two such proprietary technologies: Paknet and Mobitex. These technologies were chosen as being both representative and widely deployed. The Mobitex system in particular underpins the RAM mobile data system which has been used in a significant number of bespoke data in the field projects for public services and utility companies.

The section continues with a brief overview of the data services available through cellular mobile telephone networks: firstly, the data services available by using existing analogue voice networks (which were designed with no data service provision) and, secondly, the more recent digital services.

Paknet

Paknet is a proprietary mobile data network now wholly owned by Vodafone. Originally conceived as a replacement for fixed telephone links to electronic funds transfer from the point of sale (EFTPOS) machines, it has been expanded to include mobile users. The need to validate credit card transactions in a timely cost effective manner has strongly influenced the design of the system.

The Paknet network is underpinned by a specially developed protocol optimised for bursty packet-oriented use (each packet contains approximately 125 bytes). The protocol is based on the ALOHA protocol [Schoute,80] called dynamic slotted reservation aloha (DSRA). In DSRA, a channel is divided into slots of 27ms, mobiles synchronise with this by monitoring control information from the base station. When a mobile has data to send, it randomly picks a slot allocated for reservation requests (gleaned from the control information) and transmits its request. The request contains the amount of data that is waiting for transmission. From this information the base station can inform the mobile when it can allocate sufficient slots for the data. The random access slots can be dynamically varied based on observed packet statistics. For example, if packet sizes are short then more slots need to be provided and vice versa.

Each packet transmitted includes a 12 bit cyclic redundancy checking (CRC) and forward error correction (FEC) information. The protocol can send automatic repeat requests (ARQs) if uncorrectable errors are found in the packet. The gross channel bit rate is 8Kbps and the net data rate is dependent on the number of mobiles using the channel. The protocol supports allocation of an address to a group of mobiles which can then efficiently receive broadcast information. The mobile terminal offers a V24/28 RS-232 interface to an X.25 PAD which can be easily interfaced to fixed resources.

Paknet radio modems are generally integrated into other products to target particular applications such as Tote betting, credit card verification and alarm signalling.

Mobitex

From the outset Mobitex was designed as a proprietary mobile data network. Mobitex uses a carrier sense multiple access (CSMA) technique (commonly also employed in mobile telephone networks). Periodically, each base station sends a *sweep* message containing its identification and network parameter information. When a mobile wishes to transmit it must wait for a *free signal* message indicating that the next few slots are available for random access attempts. Each mobile who wishes to transmit must then pick a random slot from these in which to make its attempt. The first mobile that gains access will be acknowledged. The acknowledgement also informs other waiting mobiles that they must wait until the next free signal. Should two mobiles transmit at the same time, the resulting collision will destroy both messages. The base station will not send an acknowledgement, so mobiles that picked a later slot will have the opportunity to ask for the channel.

Once a mobile has received an acknowledgement from the base station, it may transmit a message up to 512 bytes in length (subject to the network parameters sent in the sweep). If the base station recognises that a long message is being sent it may send a *silence* message to keep other mobiles from transmitting. The silence is cancelled by the next free signal. Mobiles needing to transmit a sequence of long messages can request a dedicated channel by sending a *long message request* to the base station.

Parameters such as maximum packet size, traffic priority and the number of free signal messages can be varied by the base station to cope with demand. For example, when the number of mobiles increases, more random slots will be required to avoid possible collisions.

Each message is comprised of a header followed by one or more blocks. Each block contains 18 bytes (shorter messages are padded with zeros) and a 16 bit CRC. These blocks are then encoded to give FEC capable of correcting burst errors up to 20 bits in length. In the unlikely event that errors cannot be corrected, the corrupted blocks within the message can be individually retransmitted using an ARQ. Each block is sent at a gross data rate of 8Kbps, the overhead of the error detection, correction and header information bring the effective data rate down to 4.6Kbps.

The Mobitex infrastructure is organised hierarchically, a number of base stations are connected to a local exchange which is in turn connected to a higher level exchange and so on. Each level is capable of performing routing functions so only communications that do not involve units under a routers' jurisdiction have to be passed up to the next level.

Mobitex operates a connectionless store and forward architecture. When a mobile roams from one base station to another and fails to acknowledge a packet, the network will examine its neighbouring base stations and forward the message on. The mobile, rather than the network, is responsible for deciding when to handover to another base station. Each mobile must maintain a list of neighbouring base sites together with signal strength and error rates to allow it to decide when to change.

Mobitex radio modems are available from IBM on type III PC card format or integrated on the motherboard of high-end IBM ThinkPad portables.

Analogue Mobile Telephones (AMPS/TACS†)

Public cellular mobile telephone networks have long been regarded as extensions to the public switched telephone network (PSTN). It is common practice to connect modems to the PSTN to effect data transfer. Indeed, as PSTN quality has increased, advances in modem technology have enabled impressively high data rates. The cellular telephone network, on the other hand, is still primarily a voice oriented medium. While it is possible to connect standard PSTN modems to cellular channels, the significantly increased bit error rates and in-band signalling (used to control issues such as power management and cell handover) create a number of problems.

In the U.K. the two most significant cellular network providers, Cellnet and Vodafone, have adopted different strategies for handling mobile data. The Cellnet approach is that access to and from the PSTN to the mobile should be transparent. It is therefore up to the user to chose a suitable landline standard which operates a protocol sufficient to deal with the errors introduced by the mobile link. Cellnet argue that although these standards are not "cellular aware" the protocols should still be effective as the majority of data transfers are made while the mobile is stationary (reducing the data loss caused by fading and network control functions that are induced by movement).

The data arm of Vodafone, Vodata, has opted for a non-transparent solution. In particular, they have encouraged the development of a public domain protocol called the cellular data link control (CDLC [Frazer,87]). CDLC is a block based protocol based on concepts from HDLC which incorporates CRC for error detection, and both FEC and ARQ for error recovery. CDLC is based on CCITT recommendation V26bis, allowing 2.4Kbps full duplex operation (assuming no ARQs due to error conditions). Rather than forcing every user to adopt CDLC over the PSTN, Vodafone operate a gateway service known as the Vodafone Mobile Data Conversion Service (VMACS). The gateway automatically adapts between CDLC from the mobile and the landline standard of the other party (V21, V22, V22bis, V23 and V42 are supported). VMACS supports flow control to cope with the speed differential between the landline and mobile protocols.

One example of a data product for use with cellular phones is the Motorola "Power Class 14.4 Cellular Modem". Available in type II PC card form, the modem can be plugged into the data socket of MC^2 style flip phones. The modem allows fax and conventional data operation at a variety of data rates, theoretically up to 57Kbps

† North American Advanced Mobile Phone Systems standard (AMPS) and Total Access Communications System (TACS) respectively

using V.42bis compression (realistically, channel conditions will limit this to 2.4-4.8Kbps over a cellular link).

Cellular Digital Packet Data (CDPD)

Recent initiatives in the U.S. are leading toward widespread packet data services that are overlaid on the existing analogue cellular telephone infrastructure. The incremental cost of upgrading the existing base stations is far less expensive than deploying a new independent infrastructure to support data services.

The most successful of these initiatives, CDPD [McCaw,93] is supported by a number of significant cellular operators. However, the network is restricted by the geographical scope of the service provider, giving each domain effective control over interfaces between the mobiles and the network, interdomain communication and the external interface to the PSTN and other fixed networks.

Mobiles communicate with the network using 378 bit blocks which can be concatenated to form a maximum usable block size of 116 user bytes. Allowing for a synchronisation and management overhead with acceptable ARQs due to bit errors, this yields a realistic data rate of 9.6Kbps. The system is based on an access channel and a number of traffic channels, although under light loading the access channel can be used for data. Data is transferred within the supporting network using protocols that support TCP/IP, allowing easy interconnection to IP based wired services.

CDPD modems are available from IBM in type II PC card format, or can be directly integrated on the motherboard of high-end portable machines.

Global System for Mobile Communications (GSM)

GSM is a digital mobile telephone standard, initially formulated as a standard for Europe, that has also been taken up in a number of non-European countries. GSM is based on ISDN technology adapted to work in a mobile context (and consequently at a lower data rate; the basic rate of ISDN is 64Kbps).

A GSM base station operates a number of carriers each with the potential to support 8 channels (via time division multiplexing). Each of these channels covers a frequency range of 200kHz and supports a user data rate of 9.6Kbps. The standard incorporates mechanisms for splitting each channel into two half capacity channels, subject to specification of a half rate CODEC, each providing 4.8Kbps capacity. Each carrier uses the same slot structure for both voice and data which allows them to be intermixed.

Being a digital standard, modems are not required on GSM. However, it is necessary to provide data gateways to fixed networks such as PSTN. These gateways, known as *interworking functions*, provide conversion services analogous to the VMACS used in the Vodata network.

The GSM standard offers both transparent† and non-transparent data services. The transparent service, as the name implies, does not guarantee data integrity, although bit error rates are maintained at around 10^{-3} (if conditions deteriorate then handover to a better channel will occur, when none are available the connection is dropped). Lower bit error rates are possible with a corresponding reduction in data rate. Due to the TDM nature of the carrier, a fixed delay in the order of 100ms can be expected.

The reliable service makes use of a radio-link protocol (RLP) based on HDLC. The RLP provides error detection using at 24 bit checksum with FEC and ARQs. The gross data rate of a GSM channel is 22.8Kbps, allowing for synchronisation and other overheads this brings the uncorrected rate to 12Kbps. An additional 10% of capacity is reserved for ARQs, leaving a user data rate of 9.6Kbps in the presence of typical bit error rates. Both the transparent and non-transparent data services are connection-oriented, however connectionless services are under development (see below).

In addition to voice and data services, GSM offers a short message service (SMS). The SMS allows bi-directional data messages of up to 160 characters to be transmitted over the control channel. In contrast to the other services, the SMS is connectionless and provides store and forward capabilities. In addition to the point-to-point SMS, a message can be broadcast to all mobiles within a cell. This last service lends itself well to supporting information services such as paging or road traffic information.

Work is currently ongoing in extending the GSM standard to provide two new services: High Speed Circuit Switched Data (HSCSD) and General Packet Radio Services (GPRS). The HSCSD makes use of the TDMA structure of GSM to allocate more time slots to a single connection, effectively increasing the data rate. Since each voice or data connection occupies a single timeslot (data rate 9.6Kbps), the HSCSD service should offer a potential maximum of 76.8Kbps by using all 8 timeslots (the current specification will offer 64Kbps to bring it in line with ISDN). The HSCSD service will incorporate an on-demand timeslot allocation strategy to avoid a single user monopolising all the available timeslots on a carrier.

---

† Note that this use of the word transparent should not be confused with the forms of transparency provided by distributed systems platforms. In the data service context, a transparent service is one that provides no error correction. In contrast, a non-transparent service is fully error corrected.

The GPRS service will offer packet-oriented data services ranging from 14Kbps up to 115Kbps (1-8 timeslots, gross rate) by employing the same time slot grabbing techniques as HSCSD. The system is being designed to integrate with X.25 and TCP/IP networks through new interworking functions. In addition, the GPRS data services will be available at the same time as voice calls to the same handset. The first GPRS service is expected to be publicly available in late 1998.

The current GSM data service requires an interface card to enable portables to be able to access the service. One such card, made by Nokia (called the “DTP-2 Cellular Data Card”), is available on PC card type II format. The card supports the Hayes ‘AT’ command set which presents a modem-like interface compatible with most communications software.

2.2.1.4 Other Services

This section outlines some of the other wide-area technologies not previously presented. These include very wide-area coverage services such as message paging and satellite based communications services.

Paging Services

Paging systems attempt to provide communications over a very wide-area, supporting hundreds of thousands of subscribers, regardless of their location. Such systems are based on a number of simplifying assumptions: messages are unidirectional (being transmitted to the subscriber only), each message consists of a small number of alphanumeric characters and outright throughput performance is not an issue. However, performance in terms of the timely delivery of messages, low lost call rate and low probabilities of a false calls are all necessary to make the system acceptable to the users.

In the U.K. in 1975 the Post Office Code Standardisation Advisory Group (POCSAG) was formed to develop digital paging standards that would be acceptable Europe-wide. The POCSAG paging service [Telecom,78] operates at 512bps which, in paging terms, translates to 15 calls per second. Numeric pages can be sent using 4 bit codes which define 16 symbols including numerals and parenthesis. Pages can be tagged with an identifier which indicates that the message is comprised of 7 bit ASCII codes.

The POCSAG paging service is made up of a number of cells. Cells are grouped into major and minor cells according to population density. Areas of higher density are allotted more time slots than low density cells to cope with paging demand. When

a user is paged the message is transmitted to all the cells (neighbouring cells transmit the message in an alternate time slot to avoid interference).

British Telecom Mobile Communications (BTMC) operate a satellite based paging service which utilises a POCSAG code at 1200bps. The satellite service covers most of the European Community. On motorways and open road the service has been shown to deliver close to 100% of messages, although in urban and heavily wooded areas, delivery can be as low as 35%. Losses are due to attenuation of the signal caused by the low angle from geostationary satellites. A simple retransmission scheme in conjunction with a parity bit and error correcting codes help ensure message delivery. The system is well suited to use by long distance hauliers and professional travellers.

One of a new generation of satellite based paging systems is operated by PageSat called PCSAT 100. Using similar technology, the system offers a one-way data link of up to a theoretical maximum data rate of 57.6Kbps. The unidirectionality of the system strongly limits its applicability in general purpose mobile computing, though the system is well suited to broadcast information services such as news bulletins or stock market information.

Recently, a new generation of pagers have been developed which offer bi-directional communications services. This technology is currently offered by SkyTel [SkyTel,96] and Motorola [Motorola,96]. Users of the service can respond to pages via a number of pre-programmed responses. In addition, the pagers can connect to the serial interface of a portable computer, enabling general purpose messages of up to 500 bytes to be sent. A conventional pager that plugs directly into a type II PC card slot and also offers a standard LCD display for stand-alone use, is offered by Socket Communications [Socket,96].

Satellite Based Services

Since 1988 an American company (QUALCOMM) has offered a satellite based message system known as OmniTRACS. Unlike the voice oriented private mobile radio and telephone services, the OmniTRACS system is designed to provide two-way *data* communication. As a value added service OmniTRACS can provide location information for each subscriber (similar to the Global Positioning System (GPS) [GPS,92]). A typical application is to allow haulage firm dispatchers to be able to monitor fleets of lorries in transit and provide their drivers with up-to-date information. In return the dispatcher can receive diagnostic information from the lorries (for example reports of engine faults or telemetry from special equipment).

A more ambitious system, known as Iridium, is being developed by Motorola. Iridium is designed to be a digital personal communications system offering world-wide coverage. Rather than replacing conventional PSTNs, Iridium is aimed at low traffic density areas such as the ocean, sparsely populated areas and areas where personal communications are now emerging (such as developing countries). The system comprises a constellation of 77 smart satellites in a low Earth orbit which give line of sight to one or more satellites from any point on the globe. Each satellite covers 37 cells of 360 nautical miles in diameter, organised in a hexagonal pattern of concentric rings. Since each satellite travels at 7,400 metres per second, the users of the system appear static and the cells move, leading to a more deterministic handover procedure where handoffs are largely in one direction and probably to one of two cells (in a conventional cellular telephone system the user could potentially handoff to any adjacent cell).

Each Iridium channel uses 8KHz of frequency bandwidth into which voice is encoded using a voice encoder/decoder (vocoder) running at 4.8Kbps. Alternatively, each channel can run a data link at 2.4Kbps. In addition to these services the network will provide a time (GMT) and location (GPS) service to subscribers. The Iridium system is currently still under development, and consequently no products are yet available.

2.2.2 Metropolitan-Area Communication Technologies

Metricom are a company that specialise in proprietary micro-cellular packet switched networks that are designed to cover metropolitan-areas (typically the size of a large University campus). Although coverage is restricted when compared with wide-area cellular networks, the micro-cells can offer higher throughput.

Metricom's most widely accepted product to date, called Ricochet, is based on spread spectrum frequency hopping technology in the 902-928MHz band (which is unlicensed in the U.S.). The 26MHz of bandwidth is split into 160 non-overlapping channels of 160KHz each. Each micro-cell contains a pole-top repeater unit that covers a radius of approximately 1/4 of a mile. Repeaters are typically mounted on street lighting poles, giving a convenient source of power and line of sight to neighbouring repeaters.

Repeater units are wirelessly connected to form a mesh network. Repeaters connected to the wired resources are known as wired access points (WAP). There is clearly a tradeoff between the infrastructure cost of wiring more repeaters and the number of hops between WAPs. Data is transmitted at a gross bit rate of 100Kbps, though real performance is governed by protocol overhead, the number of hops

between WAPs, the coverage at the mobile's location (which is influenced by altitude and surroundings), the number of mobiles in the same cell and, if the mobile is indoors, the structure of the building.

The Ricochet radio modems are approximately pocket size and connect to the computer via a standard RS-232 serial interface. The modems are driven using a standard Hayes 'AT' command set. Two fundamental modes of operation are possible: using the Ricochet network, in which case data rates of 9.6Kbps to 28.8Kbps can be expected, and peer-to-peer mode where two modems can communicate directly at up to 40Kbps (in favourable conditions).

Subscribers to the Metricom service are charged flat fees for connection and access to additional services like the Internet and PSTN, there are no per packet charges.

2.2.3 Local-Area Communication Technologies

Wireless local-area networks offer high bandwidth communications services at the expense of area covered. In addition, the technologies are by necessity based on a wide range of robust coding techniques that enable them to operate in electromagnetically "busy" environments such as the typical office and industrial complexes. This section commences with a study of the motivations for wireless LANs, before considering the pervading standards that govern the area.

2.2.3.1 Motivation

Wireless local-area networking technologies are designed to provide network access to users inside buildings and within the immediately surrounding area. Wireless LANs have arisen from three independent development paths: cordless systems, cable replacement systems and network access technologies.

The first of these, cordless systems, are a generalisation of cordless telephone technology, which permits the user freedom of movement whilst retaining access to some fixed resource such as a PSTN line. The demand for further developments in the technology are in part due to work practice studies that have shown that sales people will often use their mobile telephones while seated at their desk in preference to their conventional phones for the added freedom of movement.

Cable replacement systems have been designed to permit office reorganisation without costly rewiring. In addition, computers can be networked in buildings where a wired network would be prohibitively expensive or not possible, such as in historic or listed buildings. Cable replacement systems are generally static and involve a non-

trivial re-alignment process if components of the system move. The essentially static nature is exploited to provide good quality high bandwidth connections with data rates comparable with fixed LANs. These technologies are generally unsuitable as a support technology for mobile computing.

The remaining category, network access systems, provide a wireless hop to a fixed backbone network such as a token ring or Ethernet. These products are designed to permit portable users who need to plug into network resources without “docking” facilities. In addition to wireless network access, a number of companies offer wireless interconnect products to form wireless bridges between wired network segments. Wireless bridges often offer very high bandwidths but like cable replacement systems allow no freedom of movement making them unsuitable for mobile computing. They are not considered further in this section.

Wireless local-area networks typically consist of base units connected to a fixed network backbone and one or more mobile units. The base units can be arranged to provide the optimum coverage in a given area. However, to give improved bandwidth utilisation and coverage, techniques often found in wide-area networks such as trunking and federation into cells are also employed in the local-area.

This section begins with a discussion of the DECT cordless access standard then moves on to a more general discussion of available wireless LAN technologies, including emerging standards and a synopsis of next generation wireless LANs.

2.2.3.2 Digital European Cordless Telecommunications (DECT)

In 1988 the ETSI started work on DECT, the European cordless communication standard. A DECT network is comprised of a common control fixed part (CCFP) to which are connected one or more base stations. CCFPs can be interconnected to form private networks spanning larger areas (multiple buildings). The mobiles in the system are responsible for initiating inter-cell handover and intra-cell channel swapping based on channel congestion, received signal strength and error criteria. The on-demand assignment of channels to mobiles is known as dynamic channel allocation (DCA). DCA makes good use of channel resources and eliminates the need for advance frequency planning. When mobiles migrate between cells the new connection is established before the existing one is relinquished, giving effectively seamless handover (particularly important for highly synchronous traffic such as voice).

The CCFP maintains databases of the location of local and foreign mobiles within its domain. Voice and data traffic are routed via the CCFP where a PBX can connect

it to external networks such as ISDN. The CCFP can also act as a gateway to data traffic on to a LAN.

DECT has been designed to provide *multibearer connections*: i.e. a range of services which support different applications. Each DECT channel is split using TDMA into 24 slots. Slots may be further divided into half slots or grouped into double slots. Each slot (or derivative) may be allocated to either uplink or downlink data, and may operate in either protected (with error detection) or non-protected mode. Slot allocation can be tailored to give varying amounts of bandwidth appropriate to each service. For example, a voice call might require full-duplex operation (equal numbers of slots in each direction), whereas a fax transmission (largely asymmetric) would require simplex operation. Services defined for DECT include PSTN, ISDN, fax, video, X.25, LAN and interworking with GSM. A summary of slot allocations to services is presented in table 2.1 (based on information presented in [Wong,95]).

Service	Synch (•)	Protected (•)	Slot Type	Slots Up/Down	Total Slots	Usable Bit Rate
32Kbps voice	•		1 full duplex	1/1	2	32Kbps
9.6Kbps fax	•	•	1 full duplex	1/1	2	25.6Kbps
64Kbps fax		•	1 full duplex 1 full double simplex	3/1	4	76.8Kbps
64Kbps video	•	•	3 full duplex	3/3	6	76.8Kbps
256Kbps LAN		•	1 full duplex 5 full double simplex	11/1	12	281.6Kbps
2B + D ISDN 114Kbps	•	•	6 full duplex	6/6	12	153.6Kbps
1B + D ISDN 80Kbps	•	•	4 full duplex	4/4	8	102.4Kbps

Table 2.1 - Slot allocations for DECT services

Continuing developments in video compression (such as H.261 and subsequently MPEG) permit videophone and video conferencing applications to be implemented over DECT wireless networks [Vaisey,92], [Heron,92].

Ericsson Telecom manufacture large scale business cordless telephone systems conforming to the DECT standard. They call their technology FreeSet.

2.2.3.3 Wireless Local-area Networks

The IEEE have established a working group to standardise the media access control (MAC) functions of wireless LANs. The standard, designated “IEEE 802.11 for wireless LANs”, wishes to promote interoperability between the network products supported by the various manufacturers. In addition, they aim to allow different networks that share the same physical location to operate without the need for

coordination (as might be found in an office building shared by a number of independent businesses).

The IEEE 802.11 MAC layer has specified a variant of the CSMA/CD protocol used in fixed Ethernet LANs. The protocol has been extended with ‘ready to send’ and ‘clear to send’ messages which are used for collision avoidance. These messages are transmitted by the base units to inform mobiles and base stations within range when data is about to be sent.

Following DECT, the ETSI have set up working groups to look at future Europe-wide radio LANs. One of these groups, the Radio Equipment and System subtechnical committee (RES-10) is drafting a standard for the High Performance European radio LAN (HIPERLAN). HIPERLAN is intended to provide data rates of between 15 and 20Mbps to the user.

Wireless LAN enabling technology can be placed into two fundamental groupings: infra-red (IR) and radio frequency (RF). IR technology can take one of three forms :-

Diffuse IR

Uses optical transceivers which are aimed at a common diffuser mounted conspicuously in a room (for example on the ceiling). Incident IR rays are diffused to bathe the surrounding area upon striking the diffuser.

Reflective IR

Rely on the optical properties of their surroundings to bounce IR rays between transceivers.

Line of sight (LOS) IR

Require transceivers to be carefully aligned and rely on a clear path being maintained between them.

Of the three, diffuse and reflective systems offer the greatest flexibility in terms of movement, LOS systems are essentially static (ideally suited for cable replacement). Directed LOS systems, such as the InfraLAN, can currently provide up to 16Mbps throughput. Diffuse and reflective systems, such as the Infralink, attain a far lower throughput typically around 40Kbps.

All IR technologies are subject to the same constraints: transceivers are only effective for distances up to approximately 25m and suffer from interference by incandescent lighting and sunlight. In its favour, IR radiation is easily contained by glass and office partitions giving easy cellular division and preventing snooping by

eavesdroppers. Use of IR technologies is currently unregulated and hence does not require a licence.

Radio frequency wireless LANs most commonly employ spread spectrum technology, though Motorola make a product called Altair II Plus which provides high bandwidth (5.7Mbps) over a range of up to 30m using microwave frequencies in the 18GHz band. However, using such a high frequency carrier, Altair has to cope with high levels of reflection and signal fading, resulting in a highly complex antenna system. The microwave technology employed is currently bulky and heavy which, coupled with the limited range, renders it unsuitable for supporting mobility.

Spread spectrum techniques are generally used in a portion of the frequency range allocated to industrial, scientific and medical use, known as the ISM band. The ISM band resides at 902-908MHz, 2.4-2.4835GHz and 4.725-5.85GHz in the U.S. and 2.445-2.475GHz in the U.K. Use of the ISM band is currently unlicensed. Spread spectrum techniques fall into two categories, frequency hopping (FHSS) and direct sequence (DSSS).

FHSS

Frequency hopping systems follow a pseudo-random frequency pattern which must be agreed by the transmitter and receiver before transmission. FHSS has the advantage that it is almost impossible to eavesdrop without knowledge of the frequency pattern and the intervals at which each hop occurs. In addition, FHSS systems are resilient to multipath fading effects which are largely dependent on frequency. However, synchronising to a new frequency each hop requires fast frequency synthesisers and hence high implementation costs.

DSSS

Direct sequence systems modulate the carrier with a pseudo-random sequence of binary bits, spreading the spectrum of the waveform. By utilising different sequences, a large number of signals can share the same frequency bandwidth. Each receiver de-spreads the signal corresponding to the binary sequence it is interested in. Other signals overlaid on the same spectrum with different codes appear as noise and are filtered out automatically. Each transmitter is allocated a unique binary sequence or code to spread and de-spread its data on the channel. Channel sharing by code (binary sequence) is sometimes known as Code Division Multiple Access (CDMA).

There are numerous examples of wireless LAN products that are based on these forms of spread spectrum technique, such as the Xircom [Xircom,96] and RangeLAN II [Proxim,96] systems. Both systems are available on type II PC card and support multicell roaming with a throughput comparable to low-end wired LANs (1Mbps and 1.6Mbps respectively).

2.3 Towards Continuous Connectivity

The preceding sections have presented a representative sample of the currently available wireless networking technologies (summarised in table 2.2). Each network type presents a unique set of communications characteristics (QoS) to end-systems. For example, the available bandwidth, latency, bit error rates, error handling characteristics and coverage offered will vary considerably from system to system.

Technology	**Available throughput**	**Range**
Disconnected	0bps	not applicable
Pager	0-512bps	huge
Satellite	2.4-4.8Kbps	huge
TACS/AMPS	1.2-4.8Kbps	wide
CDPD/GSM	2.4-9.6Kbps	European
PMR (MPT1327)	1.2-9.6Kbps	wide
TETRA	9.6-19.2Kbps	wide
CDPD	4.8-9.6Kbps	wide
Metricom	9.6-28.8Kbps	metropolitan
DECT	32-256Kbps	local
Wireless LANs	256Kbps - 10Mbps	local
Ethernet	1-10Mbps	local
ATM	25-150Mbps	local and wide

Table 2.2 - Typical throughput ranges for selected technologies

In general, two of these characteristics, available throughput (bandwidth) and range, share an inversely proportional relationship (see figure 2.1).

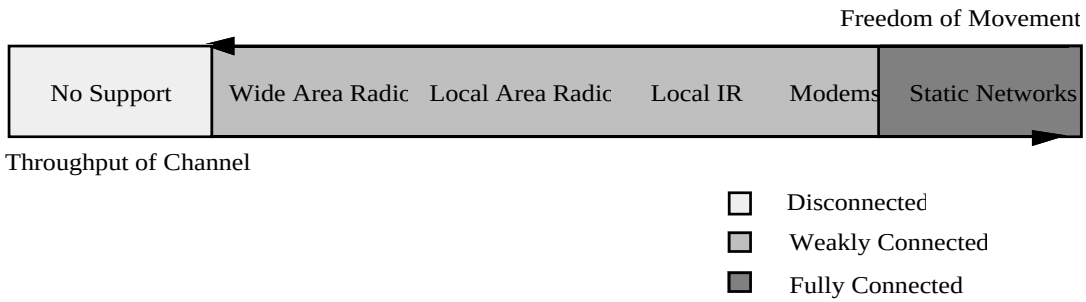

Figure 2.1 - The throughput/coverage tradeoff

Given the wide range of networking possibilities, a mobile host could utilise a range of wireless technologies to obtain continuous wireless connectivity. Indeed, a recent MosquitoNet paper [Baker,96] claimed:-

> "*We believe continuous connectivity is not only feasible but also crucial to make the most out of portable computers*".

On the surface, the selection of the appropriate network technology for a given situation seems straightforward, i.e. simply pick the service which offers the highest available bandwidth and the required coverage. For instance, when the mobile is indoors the most likely choice would be a wireless LAN. As the mobile moves outside, a metropolitan network such as the Metricom system may be available. Finally, upon leaving this domain, a wide-area technology such as GSM or CDPD could then be selected. Proponents of this approach include the Walkstation project [Hager,93] and BARWAN project (detailed in section 3.6.4). Walkstation aimed to develop an intelligent interface which interacts with the host through a standard Ethernet interface. The interface is designed to support a number of different communications interfaces, including SCSI, Ethernet, serial, parallel, infrared and microwave-radio technologies, which are seamlessly managed by the interface itself. In contrast, BARWAN supports the notion of multiple air interfaces and provides a support framework for software selection between these technologies (or *overlays*).

However, the choice of which of the possible networks to employ is complicated by two factors. Firstly, the entire quality of service offered by the network will influence the choice, not just the bandwidth. In a recent paper [Katz,96b], Katz highlighted that an important measure for comparing networks was the bandwidth per unit of coverage (specifically, bandwidth per cubic foot). By this measure an IR LAN offering room coverage of 1Mbps would offer higher bandwidth per user for five users than a 10Mbps RF LAN offering greater coverage, with fifty users. Secondly, the requirements of the application are of paramount importance. For instance, an application transmitting periodic timely messages outdoors may choose the connectionless CDPD service in preference to GSM. However, another application

may find a guaranteed GSM connection offers more predictable delay bounds for its continuous media stream.

Clearly, the applications running on a mobile within a heterogeneous networked environment will experience sudden and dramatic changes in the quality of service offered by the underlying network as the mobile roams between supporting technologies. These changes in the QoS offered by the network are of profound significance to communicating applications. Moreover, it is only by involving the application and enabling the process of adaptation that such applications will be able to continue to operate effectively. The role of application adaptation and platform support for adaptation are discussed in more detail in chapters 5 and 6 respectively.

2.4 Summary

This chapter has presented an overview of current end-system and wireless communication technologies. The significant computing power offered by these systems, integrated with widely available communications services has enabled the emergence of the mobile computing field as a major area of new research. Furthermore, this chapter postulates that through utilisation of a range of networking technologies seamless wireless connectivity can be attained.

The next chapter examines how the computer communications technologies discussed in this chapter have been utilised by the research community. In particular, the chapter focuses on system support for mobility.

Chapter 3

System Support for Mobile Applications

As highlighted in chapter 1, a mobile environment presents a number of challenges to system and application software developers, namely: variable, low and intermittent communications bandwidth, connection to and migration within, a largely fixed network and power consumption considerations given the limited battery life of most portable end-systems. This chapter presents a survey of current research and highlights two fundamental approaches to mobility: transparent and non-transparent. The chapter argues that the transparent approach is not sufficient to support advanced applications.

The research has been gathered into categories of related work. The chapter begins with analysis of network protocols (particularly within the Internet domain) and moves on to consider a range of system and application level work including RPC packages and file systems.

3.1 Network Protocols for Mobile Hosts

The main role of a network protocol is to provide end-to-end delivery of a datagram from source to destination over an interconnected system of networks. Current research is remaining true to this goal by aiming to provide transparent support for *mobile hosts* (MHs). The following sections summarise some of the more significant of these research efforts.

3.1.1 Focus on Internet

The majority of the research aiming to tackle the issues relating to host mobility have been within the Internet community. For this reason, the majority of the work presented in this section is also Internet based. An alternative, non-Internet based approach is highlighted in section 3.1.8.

Each host throughout the Internet is currently assigned an Internet Protocol (IP) address. In addition to uniquely identifying the host, the address also denotes its physical location within the network (which domain, network, sub-network etc. the host is physically connected to). Thus IP addresses are used not only as identifiers but to route all data destined for that host across the network. Therefore, when a host migrates, the IP address cannot change to reflect its new location without dramatic implications for higher level software. The solution is to separate out the identification and routing features of the IP address and manage actual host location information within the network.

There have been a number of distinct efforts to find solutions to the problems mobility introduces to the Internet. The mechanisms differ from each other in terms of how optimal packet routing is to and from MHs, how efficiently host migration is handled, how easily the mechanisms are deployed and how they affect existing hosts (backward compatibility). These mechanisms are each considered in turn below. The following sections begin with a discussion of the earliest research in to supporting mobility in the current Internet protocol (IPv4), highlighting some of the key differences between the protocols to gain a perspective on the issues involved (a more detailed discussion can be found in [Myles,93a]). The section then goes on to examine how these protocols are being fed into standardisation work to define the next generation of the Internet protocol (IPv6 [Deering,95]).

3.1.2 Sony Virtual IP (VIP)

The Sony VIP protocol allocates two addresses to each MH: a virtual and a physical IP address. The virtual address is analogous to the conventional IP address: it is configured into the host and remains constant throughout the host's lifetime and is used to uniquely identify that host. The virtual address is the only address the host is aware of; the physical IP is allocated to the host transparently. The physical address is managed by the transport layer and changes to reflect the actual location of the host.

When a MH connects to a new location, the protocol requests a temporary physical address from a local address server. The mapping between the host's permanent virtual address and newly allocated physical address is sent to the host's

home gateway. Therefore, it should always be possible to reach a host via its home gateway.

Indirecting all the host's traffic via its home gateway, regardless of where the host is physically located, can be slow and inefficient. To address this problem, Sony introduce a mechanism called the *propagating cache method* [Teraoka,93]. Each packet sent by a MH contains source and destination addresses and additional address management information including mappings from virtual to physical addresses and time stamps on each mapping. At any stage, if a Sony gateway detects that the mapping it holds is more recent than the one contained within the packet, then it updates the mapping accordingly, and vice versa. Thus, as a packet travels through gateways, the mapping of virtual to physical addresses are updated along the way. Subsequent packets are more likely to discover the mapping earlier in their journey through the gateways and optimise the route to the mobile host. Cache mappings are aged to avoid stale information being retained in the gateways. The propagating cache method is recognised as offering the most efficient packet routing to mobile hosts of any of the mechanisms presented here (assuming good Sony gateway density).

A small number of considerations have been identified by researchers for future versions. The protocol requires data link layer support to detect host migration (which is often unavailable in many systems). Once migration has been detected gateways will attempt to delete stale cached address mappings throughout the network from sibling hosts' caches. This procedure requires routers to support the Sony protocol and, as defined, does not allow for errors that can partition hosts during the process. Inevitably inconsistent mappings will be present throughout the gateways, relying on the time-outs to clear out the stale mappings. Determining the correct interval for detecting outdated mappings is difficult: too long and long breaks in communication can occur, too short and many packets will be routed sub-optimally via the home gateway.

There are a number of problems with the Sony protocol. Firstly, the protocol is not backwardly compatible; multicast is not supported and many hosts (especially those with minimal IP support, such as often found on PCs) will drop packets containing IP option fields [Myles,93b]. In addition, the processing of the options fields introduces additional processing overheads at each router encountered (estimated to be as much as 29% of the processing per packet [Teraoka,92]). Secondly, to perform optimal routing, the majority of gateways need to be updated to support the propagating cache mechanism. To support a reasonable number of migrating hosts, a large number of temporary IP addresses need to be allocated throughout the network for use by the protocol. Given the shortage of addresses in the current IP implementation, this

currently seems an unreasonable requirement. Revised versions of the Sony protocol are being considered for adoption into the next generation of IP (see section 3.1.6) as deployment of a new IP version would obviate the majority of these criticisms.

3.1.3 Columbia Mobile IP

The Columbia protocol, developed at the Mobile Computing Lab of Columbia University, deploys a small number of *mobile subnet routers* (MSRs) which conspire to create a *virtual mobile subnet* [Ioannidis,93]. From a network perspective the virtual mobile subnet appears as a real sub-network that is reached via MSRs acting as routers.

MSRs periodically multicast a beacon packet which informs any MHs of their existence, allowing them to register. Consequently, MHs can easily discover their migration upon network reconnection without data link layer support (although at the expense of the network bandwidth used in sending the beacon packets). On migration, it is the responsibility of the MH to inform its previous MSR of its nearest MSR. The previous MSR can then cache the new address for the host and forward any packets to the new MSR. Packets en route between MSRs are tunnelled through an encapsulated IP protocol known as IPIP. These messages appear like conventional IP packets and so can be routed through conventional gateways with no additional processing.

When a host wishes to message a MH, it contacts the local MSR. If the MSR does not know the location of the required host, it multicasts a location request to all other MSRs, caching the response for future use. Periodically the cached information is purged to remove stale mappings to hosts. When incorrect or out of date mappings are discovered, the MSRs communicate to help reduce the amount of unnecessary forwarding traffic.

The Columbia technique is well suited to operation over a small area or at a small number of selected sites as only a single MSR is needed at each point at which a MH may connect. However, if the system were widely deployed, the number of MSRs would soon become unmanageable. To address this concern, Columbia have developed a wide-area mode of operation called *pop-up* mode. When the MH discovers its migration (in pop-up mode this has to be from the data link layer), it aquires a temporary address and registers this with its home MSR. The home MSR can then tunnel packets directly to the temporary address. A MH capable of pop-up operation thus has to be capable of the IPIP packet decapsulation normally provided by MSRs. Although more scalable than conventional operation, optimal packet routing will only occur in pop-up mode if two communicating MHs are in the same cell, otherwise all the traffic needs to be tunnelled via the home MSR. In addition,

pop-up mode shares the same requirement as the Sony protocol for available temporary IP addresses at each connection point.

3.1.4 IBM Mobile IP

Like the Columbia system, the protocol developed by IBM requires the MH to detect and register with a special router which must be present at any point at which it may connect. This router is known as the *base station* (BAS). Each MH is allocated a home *mobile router* (MR) which must be informed once the client has registered with the BAS.

However, unlike the protocols discussed previously, the IBM protocol makes use of an existing IP protocol option called *loose source routing* to forward packets between the home and local BAS [Perkins,92]. Loose source routing enables a host to specify a list of addresses by which the packet must be routed. When a destination host receives a loose source routed packet, the IP protocol definition requires it to use the reverse route back to the source for any traffic to that destination. Thus, when a MH sends a packet to a host, if it specifies first the address of its BAS then the destination address, any replies will also be indirected via the BAS.

As with the Columbia model, the home MR is always informed of the MH's migration, enabling remote BASs to reach the host via indirection. Eventually the loose source route within a return packet will update the location information in the sending host, ensuring that the route will tend to become optimal once the MH's location settles. However, the IBM scheme currently provides no mechanism for invalidating stale routes at communicating hosts.

By making use of the loose source routing options of IPv4, the IBM scheme theoretically is both the easiest to implement and the most backwardly compatible. However, there are a number of problems associated with using loose source routing options. First and most importantly, loose source routing is rarely supported in current IPv4 implementations. As a consequence, the majority of traffic would have to be indirected via the home MR with little prospect of route optimisation. Secondly, in the event of BAS failure, the protocol allows MHs registered with that station to migrate to other BAS stations. However, responses following reversed loose source routes will attempt to establish contact with the failed BAS which will lead to a connection time-out at higher protocol layers (since the packet must be routed via all points in the loose source route).

In addition, the IP protocol requires that loose source routed packets are passed up to applications to generate the reverse route (no known applications currently support

this). In the case of UDP in particular, the communications application programmers interface (API) does not allow for this operation, so all UDP traffic will have to use the non-optimal route via the home MR. This version of the IBM protocol is unlikely to be deployed since it would require widespread support for loose source routing and additional extensions to the IP protocol to support reverse route time-outs.

IBM have subsequently attempted to address some of these problems in a modified scheme where packet encapsulation is used in preference to loose source routing. Although the new protocol avoids the problems associated with loose source routing, it requires all hosts to be modified to support efficient wide-area routing to MHs. In addition, the protocol does not contain mechanisms for optimising routes to MHs. However, some of the IBM protocol concepts have endured through collaborative efforts with Columbia to produce new schemes for mobility support in both IPv4 and the standardisation of IPv6.

3.1.5 Internet Mobile Host Protocol (IMHP)

IMHP [Perkins,94] has been developed by the authors of the Columbia and IBM protocols and, not surprisingly, owes many concepts to this earlier work. Each MH is allocated a *home agent* which maintains a *home list* of all the MHs it serves together with a mapping to their locations (if known). When a MH migrates, it registers with a local *foreign agent* (the address of the foreign agent is known as the host's *care-of* address) and informs its home agent and previous foreign agent (if applicable) of its new location. The foreign agent maintains a *visitor list* of hosts it is serving.

The mappings between care-of addresses and hosts are stored in a location cache by a cache agent (which may be part of any host or agent). Each time a host registers with a foreign agent a time stamp is associated with the location mapping to allow stale mapping detection. Each mapping has an associated lifetime which is negotiated with the agent on registration; it is the responsibility of the MH to confirm valid mappings. The IMHP management protocol allows for lazy updates and confirmations of mappings without host intervention.

The notification to the previous foreign agent is sent repeatedly until either an acknowledgement is received or the agent's mapping lifetime has been exceeded, preventing stale non-optimal routes. Optionally, the MH may simply inform the previous foreign agent that it has moved and it should be removed from the visitor list. This lack of forward pointer helps to preserve security, however sub-optimal routing to the MH will then be more likely. In addition, the MH may inform the home agent that its location should not be made publicly available to other caching agents to assist in route optimisation.

In common with the Columbia protocol, messages sent to an agent are tunnelled to the care-of address for dissemination by the foreign agent. A form of pop-up mode is defined to allow hosts to migrate to domains which do not support the IMHP protocol, although temporary IP addresses and a mechanism for allocating them to visiting hosts are then required. To help reduce communication overhead of IMHP when a host is connected to its home domain, the protocol allows hosts to switch to using standard IP protocols, providing it satisfies some simple preconditions.

Unlike the protocols mentioned so far, IMHP aims to provide some authentication of protocol messages to avoid malicious redirection of packets intended for a MH. Although strong authentication of messages would be ideal, due to overhead considerations IMHP attempts to add some security without adding any additional vulnerabilities over those already present in the current Internet protocols. The MH and home agent share a secret password which can be used to authenticate notifications of new care-of addresses for its clients. The IMHP protocol assumes that the routers on the path between the agent and the MH are trusted, so does not use any end-to-end encryption. Mapping updates are tagged with a random identifier which must be returned on the responses to be considered valid.

3.1.6 Internet Protocol Version 6

Work is currently underway in specifying the next generation Internet protocol (version 6, often known as IPv6 or IPng) to replace that currently deployed (version 4). The current Internet draft documents are attempting to address many of the issues which were not considered in the earlier releases such as multicast, improved security and authentication and, of particular relevance, host mobility.

Draft proposals for mobility support in IPv6 have been submitted by the proponents of the mobile IP extensions discussed so far, namely, VIP [Teraoka,95], IMHP [Perkins,95b] and the IBM protocol [Perkins,95a]. The protocols suggested in the proposals are strongly influenced by the earlier work outlined in the previous sections. However, some of the problems affecting the earlier protocols such as backward compatibility, the requirement for deployment at a majority of sites and non-implementation of options no longer apply. Consequently, these protocols have less reliance on encapsulation protocols such as IPIP for avoiding problems at routers and gateways. Additionally, more attention has been focused on issues such as broadcast, multicast, privacy and security.

The current drafts propose that mapping information between permanent and temporary IP addresses (called a *binding*†) of a *mobile node* can be cached at each network host (avoiding the intercession of a routing agent and thereby improving efficiency and scalability). At any point a mobile node may send a packet containing a *binding update* option which informs the *correspondent node* of a new mapping. Binding update messages are only intended for the correspondent node (unlike propagating cache updates in VIP) and so the message does not need any special interpretation at gateways (no additional processing overhead is incurred). Like previous schemes, an agent is always present in the home domain to receive binding updates and provide an initial contact point for packets destined for the mobile host. Unlike previous schemes, the draft specifies that more than one binding can be associated with a given mobile node to facilitate the transition process necessary to achieve smooth connection handover to a migrating host.

Packets are routed between mobile nodes using IPv6 routing type 0 which is analogous to loose source routing in IPv4 with two important caveats: firstly, the reverse route does not require intervention from the application layer and, secondly, the failure of intermediate routing points does not cause the collapse of the end-to-end connection. Packets routed via the home agent (before the correspondent node is aware of the binding to the mobile node) require the use of an encapsulation protocol called IPv6-within-IPv6 (like the Columbia IPIP protocol). The level of encapsulation is needed as the anti-tampering measures within IPv6 do not allow the home agent to adjust the headers of the incoming packets to reference the mobile node's location.

One of the key extensions proposed in the IPv6 draft standard is the inclusion of a *flow identifier* in the header of each message. Data packets tagged with a particular flow identifier can leave state information during their journey across the network; successive packets with the same identifier can be switched or routed depending on this deposited state. Flow identifier support is expected to allow for fast forwarding and "flow protocols" which support the end-to-end resource guarantees necessary to develop Quality of Service protocols desired for multimedia traffic.

3.1.7 MosquitoNet

The goal of the MosquitoNet project at Stanford [Baker,96] is to provide continuous Internet connectivity to mobile hosts via a variety of network technologies. The project aims to develop an architecture which supports seamless and transparent handoffs between network types. One of the preliminary

† Not to be confused with a service binding in a distributed systems platform.

developments in the project is a mobile IP protocol based on the IETF IP Mobility Support draft standard [Perkins,95a].

In common with the approaches previously described, in the MosquitoNet protocol the mobile host is always reachable via a home IP address. When the mobile host attaches to the fixed network it communicates via a *correspondent host* (CH), which could itself be a mobile host. A *home agent* receives packets addressed to the mobile host and forwards them to the current point of attachment. Forwarded packets are tunnelled inside IP packets to avoid router complications.

Unlike previous approaches, the overriding goal of the MosquitoNet protocol is to assume no mobility support in the visited network. Rather than some form of foreign agent, the host itself is responsible for unpacking encapsulated packets that have been tunnelled from the home agent. The foreign network is required to be able to dynamically provide a temporary IP address for the visiting mobile host. An automatic process for assigning these addresses such as DHCP [Droms,93] or those used by PPP or SLIP is required.

The MosquitoNet approach allows for mobile host support with a minimum level of modification to the existing IP network. However, the protocol provides no mechanisms for optimising routes to corresponding hosts, or dealing with the resulting congestion. In general the mobile host will tunnel its own responses through the home agent to make the correspondent believe it is still resident there. One possible optimisation is to make the mobile host send to the correspondent directly, forming a “triangle route”. However, these packets may require additional encapsulation as some routers will detect and drop these non-locally sourced packets as a security precaution.

Following some experimentation with implementations of the protocol, it has been observed that complete transparency is not always appropriate. For example, routing tables in a foreign network need to relate to the hosts real location for the host to respond correctly to probes from network management tools. In addition, some exposure of the network interface is required on the mobile host to allow it to determine when to switch to a different network technology.

3.1.8 Non-IP Based Solutions

The Crosspoint project [Comer,95] aims to demonstrate the feasibility of providing continuous Internet connectivity to a large community of users. The target environment for the initial prototype is a university campus with up to 50,000 mobile hosts. One of the major concerns in Crosspoint is how to deal with the vast number of

routing updates that will occur at the end of each seminar as the students move to their next lecture theatre.

The Crosspoint architecture consists of a wireless infrastructure of base stations coupled with an Asynchronous Transfer Mode (ATM) switched backbone network. Each mobile host is allocated a permanent unique IP address. The base stations are collectively responsible for tracking the migration of the mobile hosts. Viewed as a whole, the mobile hosts appear to be members of a large seamless LAN consisting of a single sub-network. Mobile hosts are connected to wired network resources through the base stations. As a consequence, host mobility is handled entirely by routing in the ATM backbone; the fixed campus network is totally unaware of host migration. A single IP address is chosen to identify the wireless interface of all the base stations in the network. This IP address is the default router for all mobile hosts.

Each base station maintains two virtual circuits to every other station; a high priority control traffic channel for routing update information and a lower priority data channel. A single routing update fits within one ATM cell which allows for efficient delivery and effectively confines the cell to the ATM network. The actual routing of packets is handled by a dedicated interface connected to every router and base station on the ATM. The interface is responsible for maintaining a host address to ATM virtual circuit binding mapping and transparently handling switching between circuits.

Unlike Mobile IP based approaches, each mobile host is unaware of the existence of multiple base stations. The host is detected by one or more base stations (if cells overlap) when it transmits a packet to the network. The base stations collude to ensure only one takes responsibility for the detected host. Radio signal strength is used to determine when to handover between base stations. No hardware support for monitoring signal strength is assumed; instead the strength is implied by monitoring the reception of packets from the mobile. For example, if the current base station misses a packet that a new base station receives, the implication is that the new station is receiving the mobile better (with greater signal strength) than the current one and so handoff should occur.

The current experimental prototype consists of three base stations, one router and an ATM switch. Each base station is a PC with both wireless LAN and wired Ethernet interfaces. The base stations act as a gateway to SparcStations with ATM switches. The SparcStations run the ATM driver and Crosspoint protocol software. Parts of the protocol execute on the PCs running the Xinu operating system. All the wireless interfaces are AIRLAN adapters made by the Solectek Corporation. Portable computers running Windows 3.1 with TCP/IP support need only have an IP address

and default router address configured to access the wireless facility. Portables have been demonstrated to maintain connectivity to the Internet and each other while roaming between base stations.

3.1.9 Analysis

The preceding sections have presented a range of protocols for managing mobile hosts within large scale networks. In particular, three early extensions to IPv4 were considered, namely, Sony's VIP, Columbia's Mobile IP and IBM's Mobile IP protocols. These protocols contain important and enduring concepts which form the basis of the IPv6 standardisation process. More specifically, the current draft standard includes an address mapping and caching strategy similar to that of VIP, the home agent, and packet tunnelling concepts from Columbia's protocol and the use of routing options similar to the IBM protocol. Therefore, although these early protocols are unlikely to be deployed outside research testbeds, their concepts will enable the next generation of Internet protocol to support mobile hosts from the outset.

The Crosspoint project is aimed specifically at providing mobility support within a proprietary metropolitan-area network. Crosspoint uses a high speed ATM backbone to manage routing information for Internet clients transparently. The MosquitoNet project, like Crosspoint, developed a protocol for supporting mobility with as little modification to the existing Internet protocol as possible. Interestingly, the MosquitoNet project identified that the transparency provided by current network protocols hindered them in their objective of providing seamless handoff between networking technologies.

3.2 Higher Layer Protocols

The approaches discussed so far attempt to provide support for mobility at the IP protocol level, enabling hosts to run existing UDP/TCP implementations potentially without modification. The following section discusses approaches which handle mobility higher in the protocol stack.

3.2.1 Indirect TCP (I-TCP)

Bakre's protocol, I-TCP [Bakre,95a], is based upon the premise that it is more efficient to handle connection-oriented (TCP) data at the transport layer in preference to conventional TCP over Mobile IP. The I-TCP protocol is built upon the Columbia Mobile IP protocol and assumes the deployment of a number of Mobile Support Routers (MSRs).

The protocol partitions the transport layer connection between hosts into mobile and fixed hops. Different protocols can then be chosen for each hop of the connection as appropriate for the channel conditions. This discussion focuses on the simple case of a mobile host talking to a fixed host, that is: two hops, one over the mobile network and one of the fixed network.

The API for I-TCP is designed to look similar to the familiar TCP socket interface. The I-TCP library essentially provides wrapper functions which interpose the MSR in connections to or from mobile hosts. Semantics are kept as similar to TCP as possible. However, end-to-end acknowledgements only apply to each hop rather than through to each host as intended.

As the mobile host migrates, the sockets at its MSR have to be migrated to the new MSR transparently. The I-TCP protocol requires kernel support to perform the following functions :-

- freeze a connected socket and capture its state,
- create a connected socket without any cooperation from the peer, establish the state of the socket and restart the connection, and
- delete (without closing) a socket.

While the migration is in progress, TCP segments in transit to the new MSR are buffered for processing once the transfer is complete. I-TCP has been shown to offer increased throughput over conventional TCP over a local-area wireless link. However, the I-TCP protocol breaks the end-to-end delivery semantics of TCP and requires that applications are re-compiled to use the wrapper library.

3.2.2 Snoop TCP

In contrast to I-TCP, the protocol proposed by Amir at Berkeley [Amir,95a], addresses the same TCP performance problems, but without compromising the end-to-end semantics. The router code at the base station is modified to cache data that is intended for the mobile client and to include a set of simple policies for dealing with acknowledgements and retransmissions. No other modifications are necessary to either the fixed hosts, mobile hosts or applications.

The base station routing code incorporates some additional functionality (referred to as the *snoop* layer) which monitors every packet that passes through the base station in either direction. Packets destined for the mobile host that have not yet received an acknowledgement are cached. If packet loss is detected, for example via detection of a duplicate acknowledgement packet, then the base station retransmits the

cached packet without passing the acknowledgement back down to the fixed host. The fixed host is therefore kept unaware of the lossy nature of the wireless link, which prevents unnecessary congestion control mechanisms from coming into play (which has been identified as the primary factor affecting unmodified TCP performance over a wireless link [Cáceres,94]).

Simulation has shown that this approach can handle a bit-error rate an order of magnitude higher than unmodified TCP over the wireless link. Currently, work on implementing the protocol over a test bed of laptops and PCs using NCR's WaveLAN is underway.

3.2.3 Mobile RPC (M-RPC)

M-RPC [Bakre,95b] is based on a similar principle to that introduced in I-TCP (see section 3.2.1), specifically the concept of an indirect protocol, i.e. one that uses an intermediary to split connections to mobile hosts into fixed and mobile connections. M-RPC has been designed to provide a standard interface based on SunRPC [Sun,88]**,** although additional support for dynamic binding to servers and disconnected operation has been suggested.

Each M-RPC binding is split into two parts, utilising UDP for the wired hop and RDP (Reliable Data Protocol) to cope with the higher error rates associated with the wireless hop. Like I-TCP, M-RPC relies on mobility support routers (MSRs) to be running on a node on the fixed network within each cell.

Before beginning an RPC, the M-RPC client must obtain a client handle, which may map on to either a particular server or a service name. Service names can be bound dynamically by the MSR to a local server providing the required service. Service name connections are re-evaluated during each migration to ensure continuation of service where possible. The MSR is also responsible for retransmission of failed RPCs which have successfully navigated the wireless hop but failed to reach their ultimate destination on the fixed network. This mechanism should help to avoid costly retries over the unreliable hop.

Migration between cells is handled by swapping MSRs; client handle state is transferred from the new MSR to the previous MSR to effect the change. The transfer of the socket state requires the same mechanism as employed by I-TCP. When a MH disconnects, for either coverage or power saving reasons, M-RPC can either store results of ongoing RPCs at the MSR and forward upon reconnection or alternatively inform the client that it may no longer use the RPC mechanism, optionally invoking an application specific handler.

3.2.4 MOWGLI

MOWGLI [Kojo,94] is a communications architecture which is designed to support mobile hosts using a public wireless data network (such as GSM, see section 2.2.1.3). The architecture, in common with I-TCP and M-RPC, is based on the concept of indirect interaction between software on fixed and mobile hosts. All data traffic between an application on the mobile host and its counterpart on the fixed network is tunnelled via an agent on the mobile host and a proxy on the fixed host gateway (known as a Mobile-Connection Host or MCH).

The MCH may act as a gateway to one or more mobile hosts. For each host a virtual network interface is created with the same IP address as the mobile node. The MCH uses this interface to act as an agent on behalf of a mobile client. Once the mobile host has connected to the network, a Mobile IP implementation is used to manage the connection. Further host mobility is taken care of transparently by the chosen data service.

The MOWGLI architecture supports a traditional BSD socket interface, allowing mobile unaware applications that use UDP or TCP to execute without modification. In addition, MOWGLI offers an API called the Mowgli Socket Interface to applications on the mobile node. The API defines some new socket attributes and operations such as prioritisation and the specification of disconnection failure semantics. Customised agent processes can be associated with the socket on the mobile node and at the MCH to introduce application specific channel optimisation.

In addition to the socket interface, MOWGLI provides two additional services. The first, the Data Channel Service, offers a reliable data transfer service between the mobile node and the MCH. Reliable stream and sequenced datagram channels are provided to underpin TCP and UDP respectively. The service also offers an API for establishing parameters such as priority, failure semantics and assigning channels into related groups. The second service, called the Data Transfer Service, manages the spooling, queuing and transfer of Data Transfer Objects. Objects can be prioritised and deferred until specified conditions are met before transmission. Attributes such as reliability, persistence, related operations (such as agent mediation or compression) and notification of successful transfer can also be assigned to the objects.

MOWGLI has been used to implement a wireless WWW browser which, through agent mediation, has been shown to achieve a 25% increase in responsiveness over the non-modified browser using a GSM telephone link. The implementation has been conducted largely on Linux, although a Windows library is also supported.

3.2.5 Loss Profiles and Probability of Seamless Communications

The work at the University of South Carolina [Seal,96] identifies two new QoS parameters designed to describe host communication requirements. The first, called a *loss profile*, is designed for applications operating in a mobile environment. This parameter allows those applications which can tolerate loss to specify at connection set-up time a preferred mechanism for reducing the bandwidth requirements of a particular data stream. This mechanism enables the bandwidth requirements for hosts in a given cell to be adjusted as users migrate in and out, enabling the available bandwidth to be redistributed.

Support for mobile hosts is assumed to be provided by mobility support stations as offered by the majority of mobile IP protocols (the need for which is highlighted by Imielinski in the overview paper [Imielinski,94a]). A contrasting approach for providing communication support for connection-oriented continuous media (particularly audio and video streams) is offered by the work of Keeton et al. [Keeton,93].

Support for loss profiles is provided by a new element in the protocol stack known as the Loss Profile Transport Sub-Layer (LPTSL). The LPTSL views the data stream as being composed of logical segments, the bounds being defined by the application generating the stream itself. Each segment is separated by the insertion of flags within the stream. It is undesirable to implement the segment filtering within the transport layer as application specific knowledge is required. Thus the LPTSL sits just above the transport layer.

The architecture describes an additional agent known as a *supervisor host* (SH) which controls MSSs on the fixed network. The SH is responsible for handling the majority of routing, protocol details and QoS management for the mobile hosts. It is the SH that needs to be made aware of how to discard information with regard to the specified loss profile. A library of “discarding functions” is provided to the source station from which it can choose the most appropriate. The discarding function is then executed by the SH when it needs to reduce the bandwidth of the communication from the source host. Typical discarding functions are clustered loss and random uniform loss, both of which are applicable to multimedia traffic such as a video stream encoded using MPEG-2 [ISO,93].

The second QoS parameter is the *probability of seamless communication*. In this context, seamless communication refers to the process whereby a host’s connections handover to a neighbouring cell without a break in service. The probability of seamless communication is thought of as the requirement of the application for seamless communication. The duration of the break in service (experienced as jitter

on the channel) is reduced by “predictive buffering” which actually means multicasting packets to neighbouring cells in case the mobile host migrates. The overhead of predictive buffering can be quite high, especially where the multicast group consists of all possible neighbours (to predict all possible movements). This parameter represents how sensitive an application is to failure to predict its movement. So, for example, if the application was relatively insensitive to jitter during handover then fewer cells need to be members of the multicast group (these are typically chosen based on some prediction extrapolated from past movements of the mobile host).

The proposed architecture for evaluating these QoS parameters consists of mobile hosts, mobility support stations and supervisor hosts running within a pico-cellular LAN [Singh,96]. The VIP protocol (see section 3.1.2) is used to support host mobility (the protocol has been modified to obtain local addresses from supervisor hosts upon migration to a cell). A single supervisor host may manage multiple cells to facilitate QoS renegotiation during migration. Packets destined for mobile hosts are cached at mobility support stations which are then responsible for ensuring reliable delivery (if required). Packets contain sequence numbers and connection identifiers to enable duplicates to be detected.

End-to-end QoS between mobile hosts is split into separate hops (mobile to fixed supervisor host). Each stage of the connection has independently negotiated QoS parameters, thus the link from the service provider can be negotiated as if there were a dedicated link between the MH and the SH, which minimises the complexity of QoS renegotiation during migration. South Carolina are also engaged in an investigation into support for reliable multicast in a mobile environment [Brown,95].

3.2.6 Cost Efficient Adaptive Protocol

The work of Lai at Monash [Lai,95] is concerned with developing a protocol to support “advanced mobile database applications”. The protocol aims to compensate for low bandwidth links and, additionally, reduce the total volume of traffic while maintaining adequate response time to applications.

The Adaptive Queuing Protocol (AQP) is targeted solely at communication from mobile hosts to stationary hosts; routes in the reverse direction are not considered. The protocol sits above the transport layer and requires a reliable transport service such as TCP. AQP batches groups of transactions sent in rapid succession into one packet to save on the header information (although it is not clear how these larger packets are affected by the high error rates typically experienced in mobile environments, particularly where ARQs are employed).

In addition, AQP operates priority based multi-level queues. The queues are drained according to two parameters: the time interval between queue processing and the time transactions are kept in the highest priority queue before transmission. When transmission occurs, the queued transactions are sent in the smallest possible number of packets (which is network dependent).

The performance of AQP is highly sensitive to choice of priority for transactions and the choice of the time interval parameters. Generally, priority is chosen based on the response time required by the transaction. The protocol adjusts the queue parameters based on observed behaviour. For example, the interval between queue processes should be reduced if the average time a packet spends in the queues exceeds a predefined threshold.

3.2.7 Rover Queued-RPC (Q-RPC)

Rover [Joseph,95] is a toolkit developed at MIT which is designed to support the development of mobile applications, or rather applications which execute on “roving” computers. The toolkit provides a mechanism called *queued remote procedure calls* (Q-RPC) to applications.

Q-RPC allow applications to continue making RPCs asynchronously even while physically disconnected from the network. The RPCs are written to a stable log to await reconnection or for the cost of communication to fall below some threshold before being played out. Q-RPC supports at-most-once delivery semantics, whereby RPCs are only considered delivered once the response has been received (it is then removed from the log). In contrast to traditional RPCs, sender or receiver failure or link unavailability do not cause RPCs to fail. The *network scheduler* is responsible for draining the log. In addition, the scheduler batches related RPCs to improve transmission efficiency and accepts application supplied priorities to enable RPC reordering.

Rover services are offered by *relocatable data objects* with well-defined interfaces that can be dynamically migrated between clients and servers. Information is shared between clients and servers using a check-in/check-out methodology; a RDO is checked out from a server, the client performs operations on the interface and checks it back to the server. During disconnections shared objects held by the client may be modified. Consistency is maintained on a per object basis using either a locking strategy or object specific conflict resolution on reconnection.

Each client machine runs an *access manager* which is responsible for handling all interactions between clients and servers. Additionally, the access manager maintains a

persistent cache of imported objects. The object cache operates a number of policies for maintaining an object's consistency with its primary copy, including uncachable, immutable, verify-before-use, verify-if-service-accessible, expires-after-date and service-call-back. Lists of required objects can be built up by individual applications to allow the access manager to prefetch working sets of objects.

Rover is currently implemented on IBM Thinkpads running Linux, DECstation 5000 workstations running Ultrix and SparcStations running SunOS. The Rover server can run as either a TCP/IP application or through the NCSA HTTPd server. The applications are written using Tcl/Tk scripts and binaries. Network connectivity is via a 10Mbps Ethernet, 2Mbps WaveLAN or dial-up lines. Example applications written using Rover include a wireless E-mail tool, a calendar application and a WWW browser.

3.2.8 Analysis

The preceding sections present a range of approaches for handling mobility above the IP layer. These techniques provide transparent mobility support which is required to enable mobility unaware applications to run in mobile environments. A number of the techniques employ indirection: that is, splitting connections between mobile and fixed hosts into wireless and wired component parts (I-TCP and M-RPC are good examples). However, these techniques suffer from the disadvantage that end-to-end transport semantics are lost by interposing intermediaries, potentially leading to transparent failures.

In contrast, the MOWGLI and Rover architectures provide APIs which enable applications to play an active role in the transmission of data and, in particular, take specific actions in the case of communications failure (although transparent operation is also supported). Importantly, these approaches recognise that complete transparency is not always desirable in mobile environments.

3.3 Application Level Approaches

The previous sections have presented a wide range of system software aimed at providing low level support for host mobility. In general, these approaches focus on support for handling the change in locality typically exhibited by mobile hosts. The work presented in the following sections concentrate on services designed to improve application performance over mobile channels (which are typically characterised by limited bandwidth and high connection set-up times). The work presented here is based on two fundamental premises, firstly, that through additional processing at the host, the volume of information that is transmitted can be reduced and, secondly, that

communication can be avoided in the first place by off-loading processing to remote sites that have improved network connectivity. These two approaches are discussed in more detail in the following sections.

3.3.1 Intelligent Communication Filtering

In low bandwidth mobile connections, particularly in a pay-per-use environment, it is important to reduce the use of the communications link to a minimum. Columbia propose an architecture which interposes an agent between a client and server to delay or filter information, thereby reducing bandwidth requirements [Zenel,95]. The architecture focuses on easy installation and management of application specific filtering mechanisms within the agent.

There are four basic types of action the agent may take :-

- Run an optimised protocol such as low bandwidth X.
- Omit unnecessary data; for example, dropping the packets generated by statistics gatherers such as `rstat`.
- Delay transmission of data to force the client to demand fetch it. For example, an editor may only require prompt delivery of the first page of a document (successive pages can wait until required).
- Use compression techniques to tradeoff processing against bandwidth.

It is likely that only a small number of packets are sufficiently self contained to allow a quick forward/drop decision to made. More commonly, a considerable level of application specific knowledge will be required to sensibly make these decisions.

The architecture consists of a number of proxy servers constructed upon Columbia's Mobile IP implementation (see section 3.1.3). The Mobile Support Routers (MSRs) used by Mobile IP are a natural place to perform filtering since all packets to and from the mobile host are forwarded via the MSR.

Non-filtered data is sent via the proxy server by constructing a virtual interface (which looks like a standard network interface to tools such as `ifconfig`). Filtered data is identified by a combination of IP address and port number.

Currently an API to install filters into the proxy server has not been developed; instead the mechanism has been hard-coded into the server. Work is underway to develop a generic filtering language which would be interpreted at the server and thus allow dynamic installation. The filtering language is expected to contain primitives such as forward, discard and so on.

3.3.2 Mobile Agents

In addition to the filtering techniques already described, further bandwidth reductions can be achieved by using migrating or mobile agents. The basic premise is that, rather than accessing data over a low bandwidth link and then applying some selection criteria upon it, the processing technique itself can be migrated to the remote site, applying the selection criteria and returning only the relevant data. Agent based techniques are similar to deploying a proxy server between a client and server which uses algorithms and filters to sift the data on behalf of the client. However, unlike proxies which often reside at a fixed site, agents are more mobile and often form a more general architecture which allows agents to dispatch and interact with further agents to accomplish their task.

3.3.2.1 TACOMA

TACOMA [Johansen,95] is a collaborative project between the Universities of Tromsø and Cornell, which aims to use agents to tackle problems traditionally the domain of distributed operating systems, such as process management and scheduling, messaging, file transfer and fault-tolerance. These agents are required to be portable across hardware platforms and hence are written in an interpreted scripting language. The architecture is generic, enabling any script with its own interpreter to be integrated (current development work has focused on Tcl/tk). The agents themselves are communicated using a distributed systems platform known as Horus [van Renesse,94]. Horus, like its predecessor ISIS, provides guaranteed group communication semantics in a fixed network environment. It is plausible however, that by modifying or replacing Horus with a suitable platform, the concepts that have been developed could be employed in a mobile environment (see section 3.6.3).

In order for an agent to perform an action based on some previous action, it must carry some associated data with it as it migrates. In TACOMA, where agent migration is expected to be frequent, the associated data is maintained as a list of uninterpreted sequences of bits. Thus no elaborate index structures need to be encoded and transferred. To facilitate efficient access to the data, local data that is non-migrating can be associated with the migrate-able state.

To allow agents to collaborate in TACOMA, they may exchange information by *meeting* each other. The meet operation is analogous to a remote procedure call in a traditional distributed programming model. The agent abstraction and meet operation are sufficient to construct more sophisticated computational operations such as inter-agent communication and synchronisation. Additional issues such as using the agents to procure services with *electronic cash*, scheduling the agents in autonomous domains and implementing fault tolerance, are also being considered.

3.3.2.2 Agents for Remote Actions (Ara)

Kaiserslautern's Ara project [Peine,96] has set out to achieve a portable and secure infrastructure for executing mobile agents within heterogeneous networks. More specifically, Ara aims to focus on system support for agents operating on mobile computer systems with wireless connectivity.

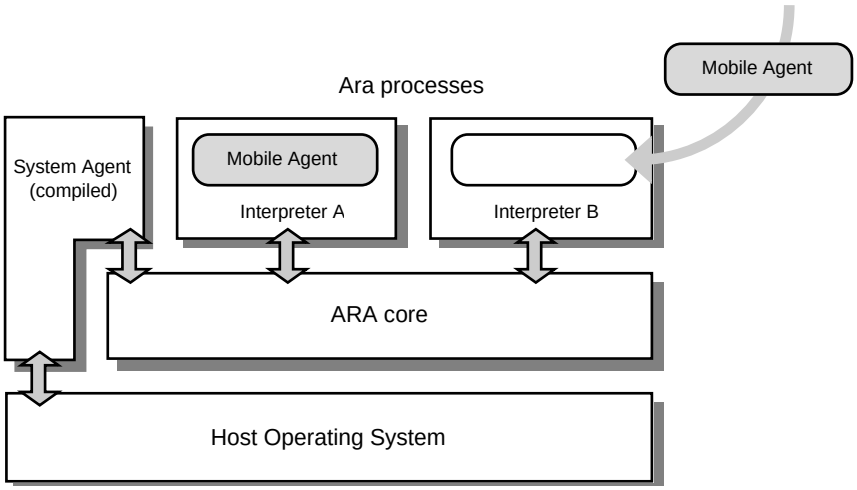

Figure 3.1 - The Ara architecture

The Ara architecture is illustrated in figure 3.1. Agents are executed by an appropriate interpreter; an agent and interpreter collectively form an Ara process. Ara processes are concurrently executed in their own independent address space by a lightweight threads package controlled by the core. The Ara core runs as a single native process on the supporting host. New interpreters need only support a set of up-call operations (for example, initialise, checkpoint etc.) and stubs to access core API functions (such as parameter checking and conversion, agent authentication and so on) in order to be integrated into the system. Currently, agents and interpreters for Tcl/tk and MACE (an interpretable byte-code translation from C++) have been integrated.

The Ara core provides agents with a set of generic services including local and remote agent directory services, named interactions with other local agents and migration control. Higher level services may be developed through the addition of system agents which are addressable through the core (system agents are compiled into architecture dependent code on each platform for performance reasons). All agent service access is via the core, enabling secure access to resources to be maintained.

The current implementation runs on networked SparcStations running SunOS 4.1 and portable PCs running Linux 1.2. The portables have local-area wireless communication via a 19.2Kbps radio modem. Future developments are expected to include improved graphical user interface access, core mediated file access, agent

persistence for improved fault tolerance, on-the-fly compilation to improve performance, and the integration of the Java interpreter [Sun,95].

3.3.2.3 Telescript

General Magic's Telescript [White,96] is an object-oriented remote programming language that has been designed for writing commercial distributed network applications comprising one or more agents. Telescript embodies three concepts: agents, places and the "go" metaphor. Places run a telescript interpreter which provides a safe controlled environment for executing agents.

Telescript is an interpreted, dynamic, object-oriented programming language. Each Telescript program (or script) is made up of one or more *classes*. Each class supports a number of *features* which are publicly visible operations and attributes. Classes may be organised hierarchically using sub-classing and *mix-ins* (a form of multiple inheritance). Features or whole classes may be *sealed* to prevent them from being overridden in descendent classes.

A number of pre-defined classes are provided in a standard library, these must be supported by every telescript interpreter. The library includes the following three classes :-

Object The Object class is always at the top of an object hierarchy.

Process Process classes provide multitasking functionality, such as multi-threading and process creation.

Agent The agent class supports the "go" operation which migrates an agent to another host.

An agent can perform the go operation self referentially. The agent must supply a *ticket* which locates the host it wishes to transfer to. If the operation is successful, the agent continues executing the next line of code at the new destination. Alternatively, if the destination host refused the agent's ticket, a trip exception is generated.

To effect an agent transfer, the sending engine must encode the agent state for decoding at the recipient engine. The agent state includes the run time state, all objects owned by the agent and all referenced classes (unless a class is built-in or known to already be supported at the destination). Consequently, an agent is completely self-contained.

Telescript engines can be interconnected to form a "telescript network". A telescript network is conceptually technology independent. Current engines support agent transfer using TCP/IP and UDP over PPP for dial-up links.

3.3.3 Analysis

The preceding sections have presented an array of complementary techniques which can be employed to aid application performance, particularly within low bandwidth environments. These techniques may be essential for improving the performance of both legacy and mobility-aware applications in mobile contexts. Significantly, both the Columbia filtering architecture and the agent based approaches recognise the role of application specific knowledge in the handling of bandwidth minimisation techniques. In the first case, through a dedicated API and, in the second, through the factoring out of application specific code into the agents themselves.

3.4 File Systems for a Mobile Environment

This section reviews work on mobile file systems, i.e. file systems designed to provide a standard file system interface which allow mobility unaware applications to run without modification in a mobile environment.

3.4.1 CODA

The CODA file system [Satyanarayanan,90] is a highly available replicated file system developed from earlier work at Carnegie Mellon on the Andrew File System (AFS) [Satyanarayanan,85]. AFS is made up of stateful servers and one or more untrusted clients. The servers are physically secure, only run trusted software and are maintained by professional staff. The clients locally cache whole files from the servers; the consistency of cached files is maintained by the servers who call-back the client whenever a cached file becomes outdated (these notifications are referred to as *call-back breaks*).

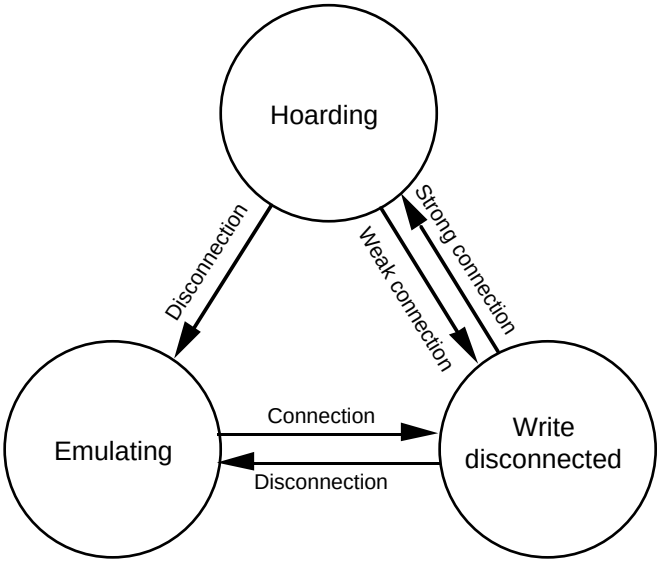

Figure 3.2 - The CODA state transition diagram

In CODA, availability is increased by replicating volumes (portions of the file space) over a volume server group. While the client is connected to the servers (in the *hoarding* state, as shown in figure 3.2) the local cache is loaded with files that are prefetched from the user's working set either periodically or at the user's request (typically before manual disconnection). The files prefetched are determined by the user's hoard database file which contains file and directory paths together with associated priorities. The database is constructed using file access traces and explicit hints from the user.

If the client cannot reach any of the volume server group or disconnects from the network altogether, the client temporarily becomes a replica site. The file system moves in to the *emulating* state where it optimistically allows operations on the files in the cache to continue as if the servers were still available [Kistler,91] (providing the file is not already known to be inconsistent). All the file operations are logged to a local update log.

When a server becomes available the client moves into the *write disconnected* state [Mummert,95]. Operations are still logged as when "emulating". However, the update log is reintegrated to bring the server replicas up-to-date. Only when all of the logged changes have been replayed and the connection to the servers is of a suitably high bandwidth will the client move back into the hoarding state. If the connection is via a low bandwidth connection (often called "partial connection") the client can use the link to receive server consistency guarantee call-backs, begin to reintegrate the update log in the background (known as trickle reintegration), service cache misses in selected cases and write back files to make room in the cache if it has become exhausted.

Before a cache miss is serviced, user intervention is sometimes required. For example, servicing a cache miss on a 1Mb file will take over 20 minutes at 9.6Kbps, so giving the user a choice about whether to fetch it or not is clearly important. CODA defines a term known as the *patience threshold*, which is a logarithmic measure based on the user assigned priority of the file in the hoard database. User intervention on cache misses is only sought if the estimated time to service the request exceeds the patience threshold.

Over prolonged disconnected periods the update log can get very large. Research has shown that a large proportion of the files are transient, existing only for a short period of time. Logged accesses to transient files such as these can be optimised from the log. For instance, the updates to a file which is subsequently removed before reintegration can be removed from the log. The benefits of log optimisation need to be balanced against the speed of update propagation in write disconnected mode.

Specifically, if the update is propagated too quickly, the optimisation routine may not have enough time to discover that the update was made unnecessarily. CODA operates an *ageing window* policy which delays write-backs long enough for optimisation to take proper effect (traces have indicated the current value of 600 seconds to be a suitable compromise).

During the reintegration process inconsistencies may be discovered. To help preserve transparency, application specific conflict resolvers can be employed to automatically reintegrate these conflicting copies (in practice roughly 80% of conflicts are handled without user intervention). If a conflict cannot be automatically resolved, a fake directory is created that has the same name as the file and contains the conflicting file replicas. The user must resolve the conflict between the replicas manually. Dependencies between file types and specific resolvers are defined in a hierarchical system of rule files, akin to UNIX Makefiles. The resolver code is executed on the client machine, as CODA servers must only ever run trusted software. To avoid a number of potential security attacks possible with the resolver mechanism, some simple defences have been employed including process permissions, trusted resolver directories and forcing manual resolution. To limit the effects of badly coded resolvers which tie up system resources, each resolver is only permitted to execute for a certain amount of time (which is user configurable).

3.4.2 Odyssey

When mobile clients access remote data in a wireless network they can expect to experience a wide of variation in network quality. Odyssey [Noble,95] is a set of extensions to UNIX to provide support for mobility. These extensions are largely at the system call level and internal to the operating system. Odyssey presents applications with an API that allows it to retrieve and present data with varying degrees of fidelity. This is termed application-aware adaptation.

The Odyssey platform is a generalisation of previous work on the CODA file system (described above). Odyssey, like CODA, subscribes to the view that resource poverty, physical vulnerability and scalability concerns of mobile hosts require a client-server architecture where fixed servers maintain repositories of data and mobile clients are merely caching sites.

In the Odyssey model, the operating systems' role is to sense external events and monitor and allocate resources. Applications should then take responsibility for using the information and resources to adapt to changes. For example, an application accessing video data may degrade the quality of the image if network bandwidth is reduced.

Data is stored in *tomes*, which are conceptually similar to volumes in AFS and CODA. In addition to normal volume contents, each tome is allocated a type or *codex*. Each codex identifies a unique combination of characteristics and policies such as naming, consistency, degradation policies to reduce bandwidth consumption and caching strategies. Like CODA volumes, tomes are the unit of replication and must be stored entirely at each replica site. Although tomes are attached at mount points, a tome need not be organised hierarchically. For example, a tome containing SQL might be named associatively. Data of a particular type must be stored in the correct tome, i.e. video data must be stored in the video tome.

The Odyssey API allows an application to negotiate for resources and receive notification when these go outside a "window of tolerance". Resources may be generic to all tomes (such as disk space, network bandwidth or battery power) or codex specific. Once an application receives a notification from the platform, it must take corrective measures and negotiate a new window of tolerance. The interface is similar to the signal interface of UNIX. The upper and lower bounds of the window are passed to the API with a resource identifier and a call-back function to invoke, should the resource transgress these bounds.

The *viceroy* is an agent within Odyssey which is responsible for monitoring the generic events and resources. In addition, the viceroy handles functions common to all tomes such as authentication, location and administration.

Type-specific monitoring is provided by codex specific *wardens*. Each warden maintains its own cache, naming scheme and resource management specific to its type. Ultimate resource control rests with the viceroy which acts as a coordinating entity between wardens. The architecture is extensible allowing new wardens to be added to manage new storage types. Implementation of the architecture is currently underway.

3.4.3 LittleWork

The aim of the LittleWork project [Honeyman,91] is to develop a mobile computing environment identical to that of the office, using only 'off the shelf' components. Consequently, to maintain compatibility with existing file servers, LittleWork's file system has been implemented on top of vanilla AFS with no modifications made to the file servers.

Unlike CODA, the LittleWork file system is not based on a replication paradigm. The project team have observed that replication does not generally improve availability in current mobile environments since typical mobile computers

communicate with a fixed network via a single mobile air interface and thus fixed resources are either available or unavailable depending on the wireless link.

The AFS client software on each mobile host has been modified. The first set of modifications involved the addition of congestion control into the AFS RPC mechanism (called RX). These modifications have been sent back to the implementer of AFS to remain faithful to the goal that components should be off the shelf.

Connection and disconnection is currently controlled by the command line instructions *disconnect* and *reconnect*. In common with CODA, operations in disconnected mode are allowed to continue optimistically on locally cached files. Each operation is logged to allow reintegration and conflict resolution when connection is re-established. Whole file caching (the mechanism initially used by AFS) simplifies cache management and conflict detection and resolution considerably. However, the most recent version of AFS caches files in blocks of 64K. The LittleWork file system has upped the block size to 1Mb, since the majority of important files are expected to fall below this threshold and will thus be entirely cached.

The cache is “pre-heated” by tracing operations while in connected mode. No explicit user intervention, such as hoarding, is required. Although it can be argued that a single trace of program execution will not yield all the files that may be used, over time the minimal working set will be constructed. In contrast, CODA’s hoard database can lead to large apparent working sets containing unused data. For example, a user might specify in the hoard database that they require the X windows system. The hoarding process may then cache all the associated X libraries and binaries including a considerable number that are rarely used. There is also the danger that the cache will be “overheated”, filled with so much information that some has to be discarded before a cache miss can be serviced.

Logged operations can be classified into those that will and will not cause file conflicts (described as mutable and non-mutable operations respectively). Non-mutable operations are still logged to allow detection of operations which have been conducted on stale information. As with CODA, a considerable amount of log optimisation is possible. In contrast however, the LittleWork optimisation process is user triggered, rather than running as an ongoing background process.

When the client reconnects to the network, a background process begins propagating updates to the server files. Directory conflicts can be resolved automatically. In the case where a directory has been deleted at the server that used to house a cached file, the file is placed in the *orphanage* for user intervention. All file

conflicts are detected and flagged for user resolution (there are currently no automatic resolution techniques).

In addition to fully connected and fully disconnected modes, LittleWork supports a partially connected, fetch-only mode for low bandwidth environments. In this mode, operations are conducted upon cached files as when disconnected. The main difference is that in fetch-only mode, cache misses can be serviced to a limited degree using the link. Additionally the server can offer improved cache consistency by using the link to offer AFS style call-backs. It is worth noting that the client may not always want the most recent version of a file. For example, should the X window library files be updated, the current cached versions will probably suffice until full connection is re-established.

The partially connected mode is supported by a version of Jacobson's compressed SLIP [Jacobson,90] (CSLIP) called Lottery SLIP [Huston,95] designed to improve response times for interactive traffic in the presence of file system activity. Lottery SLIP operates three independent priority queues (one more than CSLIP), corresponding to interactive, mid-priority traffic and file system traffic respectively (in high to low priority order). Messages are assigned to a queue on a per message basis depending on the destination port in the message header. Messages are drawn from each queue using a lottery scheduling algorithm which offers probabilistic guarantees of fairness and service [Waldspurger,94]. Lottery scheduling works by allocating a number of tickets to each item that desires access to a given resource. The higher the priority, the greater the number of tickets the item is allocated. When it is time to schedule the access to the resource a draw is held, the holder of the winning ticket is allowed to use the resource. Thus, the exact number of tickets chosen determines the rates at which messages are drawn from each queue.

3.4.4 SEER

SEER is a predictive cache whose aim is to automatically identify (and predict) the working sets of a particular user and hoard them prior to disconnection [Kuenning,94]. SEER is implemented over FICUS [Guy,90] (or RUMOR, a user space version of FICUS) which is a file replication platform layered on top of NFS. SEER, FICUS and RUMOR have all been developed at the University of California, Los Angeles.

FICUS has been designed to be a very large scale, highly available file system. As a consequence, the file system has been designed to minimise global state and to avoid assumptions about availability (as any large scale network will suffer from partitioning due to failure). Like CODA, FICUS achieves availability through

replication. In contrast to CODA, files are shared in a peer-to-peer hierarchy rather than a client-server structure.

Unlike most file systems, FICUS permits volumes from remote servers to be mounted at arbitrary *graft points* in each host's namespace. Remote volumes can be mounted automatically (known as auto-grafting) when a graft point is encountered while traversing the file hierarchy. Unlike traditional tree structures, a volume may be grafted on in multiple places in the namespace forming an acyclic directed graph.

FICUS itself is comprised of a stack of layered modules, each supporting identical symmetrical interfaces (in and out). Although the project team have sometimes found these interfaces constraining, the modular architecture allows a high degree of configurability as layers can be added or removed simply and efficiently.

Semantically, the file system supports *one-copy availability* which allows any copy of file data to be updated without requiring access to a particular copy, or number of copies. Although permitting a high degree of availability, one-copy availability offers weaker consistency than alternative schemes such as primary copy, voting, weighted voting or quorum consensus.

Each file has an associated version vector which enables the *reconciliation service* to discover update conflicts. FICUS must preserve *no lost update* semantics. Directory conflicts are automatically resolved and, as with LittleWork, an orphanage metaphor is used when remove/update conflicts are detected. Updates to files and directories are propagated to all available replica sites to maintain consistency. Like CODA, application specific resolvers are used to resolve write conflicts (the resolution problem is more general than that found with CODA due to the peer-to-peer architecture and version vectors).

The aim of SEER is to automatically determine the working file sets corresponding to user tasks based solely on observations of user behaviour. These sets are used to stock the cache to lower the probability of catastrophic cache misses and, in the event of a miss, to have sufficient alternative working sets cached to allow the user to change to a secondary or tertiary task. SEER relies on the assumption that a given working set is both predictable and sufficiently small to fit in the mobile machine's cache.

SEER defines a term called an *attention shift* to describe the process of a user turning from one task to another (one working set to another). The process of determining the working set is further complicated by issues such as process concurrency and attention shifting between sets which share common members.

To attempt to establish relationships between files (and hence their membership to a given working set), SEER calculates a measure known as *semantic distance*. The distance between two files *a* and *b* is defined as 0 if the files are open simultaneously, otherwise it is the number of files opened between *a* closing and *b* opening. The file distances are further clarified by removing accesses to common utilities and libraries such as `cc` and `libld.a`. Since distances could potentially be calculated between every pair of files, the predictive cache could quickly become too expensive in both computational and storage terms. Statistics are kept on the 10 semantically nearest files to the last 100 most recently accessed.

SEER is implemented in three parts: the *cache manager*, *correlator* and *observer* processes. The cache manager is implemented as a FICUS module. The module gains predictions by talking to each users' correlator process which in turn gathers semantic information from observer processes on each host. Initial implementation results indicate that the SEER system is quite effective for minimal system overhead. Further work is expected to examine the impact on SEER of introducing mobility support into FICUS.

3.4.5 Jetfile

The Swedish Institute of Computer Science (SICS) are developing Jetfile [Grönvall,96]: a highly scalable distributed file system designed specifically for operation in wide scale heterogeneous networks (such as the Internet). The majority of current distributed file systems rely primarily on unicast communication which places inherent limits on scalability. In contrast, Jetfile makes extensive use of the multicast communication paradigm (provided by either network hardware or, in the Internet domain, SRM).

The inherent location transparency afforded by multicast enables clients to obtain files without necessarily knowing the location or identity of the provider. A client requiring a particular file (or portion of a file) joins a multicast group hashed from the file identifier and multicasts a *repair message*. Any client or server within the group is capable of multicasting the file in response to the request (clients thus contain traditional server functionality and are referred to in Jetfile as *file managers*). By *snooping* on activity within the multicast group, file managers avoid acknowledgement implosion and, in addition, may be able to intercept relevant updates or meta-information.

To maintain consistency, each file has an associated version number which is updated each time it is modified. When a file is opened for writing, the appropriate *versioning server* for the file must be contacted. The update to the file can continue

optimistically while waiting for the new version number. The versioning server acts as a serialisation point, ensuring that the same version number is never assigned more than once and, additionally, enabling write-write conflicts to be detected (application specific resolvers may be then be used). File managers snooping the multicast group will detected the version request and corresponding repair message and realise their current cached versions have been invalidated (the version messages can be thought of as multicast versions of the call-back breaks in AFS). In addition, managers periodically ask versioning servers for version tables to verify their cache contents (these are of course multicast and will be intercepted by other interested parties). Version tables are equipped with *time-to-live* fields to enable invalidation of stale information.

For reliability, a master copy of every file is kept on a *storage server*. Jetfile operates equivalent sequential write sharing semantics to the popular NFS and AFS file systems, i.e. operations are made on the version of the file that was opened (it is not patched with updates as they occur), and writes are flushed back to the server once the file is closed. Changes to the file must be propagated back to the server before the manager leaves the relevant multicast group, otherwise the new version of the file will have no server. The Jetfile namespace is organised into a hierarchy of organisations, volumes, directories and files (directories are simply tables in ordinary files). Versioning and storage servers are associated at volume granularity.

To cope with the vast dynamic range of network QoS present in modern and emerging wide-area networks (optic fibre to wireless links), the file system features a range of adaptive prefetch and write-back policies. For example, in high bandwidth networks, large portions of a user's working set are prefetched to minimise the round-trip times of accessing the files. Furthermore, other members of the multicast group will be able to snoop the working set without incurring the propagation delay. The prefetching process operates as a background process. Once a file has been fetched, associated files are scheduled for prefetching. Working sets are stored as meta-data with each file and are predicted based on observed file access patterns. The most passive form of prefetching is simply to join the appropriate multicast group in case the information can be snooped.

In contrast, in a low bandwidth network, particularly where costs are associated with data transfer, the antithesis is true. The minimum amount of data the user requires to begin work would be prefetched. For instance, prefetching the first page of a document in case the remainder is never required. In addition, all files might be written-back at the end of an editing session to enable the updates to be batched and reduce the time spent connected to the network.

The mapping between files and multicast groups is not necessarily one-to-one. This property may be exploited to provide a range of services at different bandwidths, enabling clients to join the group that is most appropriate to their network capability. For example, a server could provide a file to low bandwidth (9.6Kbps), intermediate (10Mbps) and high bandwidth (100Mbps) groups at different levels of detail.

The Jetfile prototype runs on a the JetStream gigabit local-area network and on the Internet multicast backbone (MBone). In addition, Jetfile is designed to take advantage of the multicast facilities that will be available in IPv6.

3.4.6 Disconnected Operation Cache (DOC)

DOC [Huizinga,94] is a cache based architecture which supports optimistic whole file replication and emulates continuous server access whilst disconnected. Although based largely on concepts from the CODA and LittleWork file systems, the target domain for DOC is MS-DOS/Windows (although the intention is to support heterogeneous networks and servers). DOC is under development as part of a wider DOC project at California State University, Fullerton.

The designers identified the following requirements :-

i) The server should be unaware of the modifications required to support DOC, implying a client only solution.

ii) Communication to servers must be possible by any available interface, be it serial, parallel or via PC card.

iii) Disk accesses must be reduced to maximise battery life.

iv) The file system API must not be changed on the client, enabling DOC to remain transparent (although an extended cache API may be required).

One of the primary goals for DOC is to offer the additional partial and full disconnection functionality without introducing additional user perceived latency while the user is connected. Existing file servers make extensive use of caching techniques to improve response times, hence DOC caches to RAM in preference to local disk. Hooks have been placed to trap System Management Interrupts (SMIs) that are triggered on power cycles to checkpoint the cache to disk.

To reduce overhead, DOC uses a three layer hierarchy of caches: a primary cache in 64K of base memory, a secondary cache in 31Mb of extended memory (XMS) and lastly the local disk (if necessary). Only the primary level is used to communicate

between the cache, server and application which reduces the overhead of accessing XMS. The cache is maintained according to a least recently used strategy.

When the client is in connected mode, all modifications to a file are written back to the server when the file is closed. Once the client is disconnected, the connection to the server is emulated; allowing operations on the cached files to continue. Like CODA and LittleWork, file updates are logged to the local disk while the client is in disconnected mode. The validation of the current cache contents is performed by the client (there are no AFS style call-back guarantees).

DOC supports the same file access semantics as the Novell file system, which simplifies disconnected file access. Normal files (non-shared) are only permitted to be opened for reading or writing by one client at a time. Files tagged as shared may be opened by multiple readers but only one writer, providing any modifications are made immediately available to all of the readers. DOC caches only non-shared files, which forces shared file access to be conducted only whilst fully connected. Write/write conflicts are still possible as non-shared files can be updated by both a disconnected client and a client connected to the server simultaneously. Conflicts are detected during cache validation using time stamps which are written to the update log. The user must choose how to resolve update conflicts by selecting between the client version, server version or generating a new file. Currently no automatic conflict resolution techniques are employed.

3.4.7 Intelligent File Hoarding

Intelligent file hoarding [Tait,95] is an attempt to improve on the existing cache hoarding approaches of CODA, SEER and DOC. In addition, the new hoarding strategy builds upon earlier work on developing a file system for mobile computing [Tait,93]. Prior hoarding approaches use some combination of *spying* (user traces) and *hoard profiles* to determine which files should be prefetched into the cache before disconnection. Intelligent file hoarding uses a technique called *transparent analytical spying* to determine users' working sets.

The aims of this technique are :-

- to automatically detect working sets,
- to provide the concept of *generalised bookends* (which delimit independent user defined tasks),
- to present working sets to the user using a convenient graphical user interface, and

- allow the user to manually trigger the hoarding process.

The first aim, detecting the working sets, is based on traces of the user's file accesses observed by the IBM OS/2 'Mobile File Sync' installable file system (which is functionally very similar to DOC). At hoard time, trees representing the inter-file relationships are constructed from process identifier information in the user activity log. The trees can be expanded over multiple program executions to get a more informed picture of the libraries and sub-programs which may be used.

The trees can contain references both to application and data files, which would lead to unnecessary caching of data files that were used during the last traced executions of the applications. The hoarding process uses a heuristic to distinguish between the two file types. The first step relies on the fact that OS/2, like DOS, enforces the use of file extensions to determine content type. For example, the extension `.EXE` implies an executable file. The next step is an inference based on the file's location. If a suspected data file is located under a different parent directory to the application then it is more likely to be data than another file under the same parent (as applications and components are typically co-located). The final factor in the heuristic is based on the modification time stamps of the files; data files are more likely to have been more recently modified than application files, so if a parent and a child file have not been modified for *m* months and their modification times are within *u* months, then they are considered part of the same application. Currently, *m* and *u* are set to 3 months and 1 month respectively.

The generalised bookends allow the user to specify the beginning and ending of a period of activity; this effectively assigns a name to a portion of the file access log with which to distinguish distinct working sets for different tasks.

3.4.8 Bayou

Bayou [Demers,94] is a replicated weakly consistent storage system, developed at Xerox Parc, which provides high availability by allowing users to read and write any data replica. Rather than providing transparent support for an existing file system, Bayou focuses on supporting applications which are aware that they are reading weakly consistent data and that their write operations are likely to conflict with others at some point. Furthermore, application specific conflict resolution techniques are supported whose eventual aim is to achieve replica consistency. Bayou has been designed to support a plethora of non-real time collaborative applications, such as shared calendars, E-mail and document editing.

Each set of data (known as a *data collection*) can be replicated at multiple servers for availability, a client needs access to only one server to access the collection (a server and its clients may co-reside on the same host). Client applications may interact with any server through a programming interface supporting the operations *read* and *write*. The client may read or write to the data without waiting for it to propagate to any other replica site. *Session guarantees* can be provided to reduce client observed inconsistencies when accessing different servers. Upon writing, a server maintains an ordered log of each write and a unique *write identifier*. Local conflicts are detected and resolved immediately and any resulting changes to the data are made immediately available. Periodically, a pair of servers exchange write information in *anti-entropy* sessions. Eventually, assuming that servers are not permanently partitioned, all writes will propagate to all servers. As conflicts often occur at a site remote from the application, resolution must be possible automatically.

Bayou supports two mechanisms for detecting and resolving write conflicts, *dependency checks* and *merge procedures*. The dependency check, which consists of an application specific query and expected result, is supplied as part of the write operation. If the dependency check does not yield the expected result, then a conflict has occurred and the application supplied merge procedure is followed to perform the resolution. If a merge procedure cannot resolve the conflict, it is flagged for user intervention using a merging tool. Merge procedures can be considered more general than application specific file resolvers as they can accompany, and take specific action for, a single write operation.

Eventual consistency over the replica sites is achieved by guaranteeing two properties: writes are performed in the same well defined order at all servers; and conflict detection and resolution is a deterministic process. When a write is received by a server it is assigned a logical time stamp and initially labelled as *tentative*, which persists until it can be *committed*. A write can be considered stable at a particular server once all writes that should be executed before it have been played at the server. Non-stable writes can potentially be committed then unrolled during the anti-entropy process if causally earlier writes are discovered.

To speed the rate at which all writes become stable, an explicit *primary commit* scheme can be employed. Explicitly committed writes take precedence over any tentative writes and are propagated during the anti-entropy process to the other replica sites. Using primary commit avoids the expense of accumulating the quorum of servers required by more elaborate schemes, which lends itself to an environment which experiences frequent periods of disconnection.

The current implementation of Bayou is written in ANSI C on SparcStations and Linux PCs. The merge procedures are written in Tcl. Communication between clients and servers is via ILU, a platform independent RPC package also developed at Xerox Parc.

3.4.9 Analysis

Mobile file systems, such as those described above, play an important role in supporting legacy applications in mobile environments. By weakening file consistency and offering optimistic access to cached files, the approaches offer demonstrably effective file systems in the face of disconnection. The more recent approaches such as Jetfile and Bayou, which have been designed with mobility in mind, also use this optimistic premise (although Bayou is specifically designed to make potential inconsistency visible to applications). Interestingly, in generalising the concepts of CODA, the Odyssey team have found that an API is required to determine when and how bandwidth minimisation techniques should be applied (moving toward a less transparent approach).

3.5 Mobility in Open Distributed Platforms

There has been a limited amount of work investigating the impact of mobility on open distributed platforms. In this section two are considered: mobile DCE and mobility within the NOTUS architecture.

3.5.1 Mobility Support for DCE

The work of Schill et al. [Schill,95] at the Technical University of Dresden, aims to develop a supporting architecture within DCE which enables applications to operate in a mobile environment. In the mobile DCE model, a system is described in terms of a set of logical domains. Each domain comprises one or more sub-networks, which in turn contain mobile or fixed hosts (or *stations*). A *domain manager* in each domain accounts for all member stations within the domain, provides brokerage services for the stations' services and resources, contains an abstract network representation of the domain and keeps a potentially changing list of peer domains. Each station must be a member of a single domain only. Domain managers periodically exchange hints about the services and resources offered by remote domains. A mobile station must register (or deregister) information on the services and resources provided on the mobile on joining (or leaving) the domain.

Resources in the system are classified according to the following criteria :-

State Resources are qualified either as possessing state (*stateful*) or not (*stateless*).

Access If a stateful resource may be modified by a single application only it is designated *stateful-static*. Alternatively, if the resource can be modified concurrently it is called *stateful-dynamic.*

Emulation A stateful resource is termed *emulatable* if it can be synthesised (perhaps with a subset of the functionality) from cached data.

Volume Resources have an associated course grained measure of data size.

Applications access the station manager through subsystems which encapsulate normal DCE and OS functions, such as RPCs and file access primitives (a direct API is also available). The wrapper libraries pass operations to the station manager for service via local procedure calls. The station manager mediates access to resources and, if the service becomes unavailable, may transparently employ one of the following mechanisms :-

Full caching Allow the application to continue working from cached data.

Remote domain Access an equivalent service from the remote domain.

Remote access Access the required resource in the home domain.

Emulation Supplement cached data with (typically a subset of) application functionality through emulation.

A semi-transparent mode is also provided, in which the station manager presents the recommended alternatives to the application for selection.

The station manager makes the decision on which of these mechanisms to employ according to information stored on the application behaviour, resource and service requirements. Table 3.1 illustrates the basic decision policies for resource access over a range of networks.

Connectivity	Kind of resource			
	Stateless	Stateful-static	Stateful-dynamic	Emulatable
Disconnection	Remote domain	Full caching	Full caching (manual reintegration)	Full caching with emulation
Wide-area Wireless	Remote domain	Full caching	Remote access	Full caching with emulation
Medium bandwidth	Remote domain/access	Full caching	Remote access	Remote access
LAN	Remote access	Remote access	Remote access	Remote access

Table 3.1 - Resource management decisions

Each application's behaviour is described by a hierarchical state machine which enables the station manager to select equivalent services. The state machine may be in multiple states at the same time to represent concurrency and, additionally, may comprise multiple disjoint automata to represent non-deterministic state transitions.

The station manager includes a modified form of DCE/RPC which may rebind ongoing RPCs to alternative services. In addition, the RPC operations may be logged for later reintegration. If the substitute service does not provide the full range of operations, the RPC enables emulated services to make up the remainder (if available).

The basic domain architecture and interchange protocols of mobile DCE have been implemented under OSF/1 on DEC Alpha workstations. Work is underway to port these components to Windows NT. Early results indicate that, even with small example programs, complete distribution transparency was neither possible nor desirable in mobile environments. In addition, high level descriptions of the network and application behaviour were necessary to support semi-automatic resource management under this scheme.

3.5.2 Mobility in a Trading Environment

The NOTUS architecture [Pope,95] is based on the premise that performing a traditional connection handoff as found in IP based mobility solutions will cause all the existing IP connections between the migrating host and the original fixed host to be reconstructed. However, in a mobile environment, which may make extensive use of replication, it may be possible to find the same service from a local server in the new cell. A handoff which is reconstructed to a local service rather than the original one is considered a *traded handoff*.

In order to provide traded handoffs, more abstract information is required than the network level address. Applications are structured in terms of modules. Each module may support an interface which is an instantiation of an abstract data type. A traded handoff is achieved by renegotiating each service the client uses at the new site using properties of the interfaces.

In addition to service migration, a checkpointing package is provided which allows applications to migrate in response to network QoS changes. Application migration reduces RPC latency between a client and server or can provide a new service to a base station to avoid a client having to rebind to the original server. The checkpointing package can be used to facilitate application migration across host architectures.

When an application migrates, the initiating module checkpoints itself then informs its local name service or *trader*. The trader informs the other modules of the application which have advertised themselves as checkpointable to save their state. The trader then accepts responsibility to start a remote version of the application and transfer the checkpointed state to it. The remote trader triggers the modules to restart using the transferred state.

The architecture is underpinned by support for traditional or *flat handoffs*. A virtual network layer provides a mapping between a virtual and physical Berkeley socket interface. When migration occurs during an RPC, the physical connection is re-established to a new physical socket; the client will not receive the reply to its request which will cause a retransmission (the time-out interval can be avoided at the expense of migration transparency). Similarly, the server may receive a duplicate request from the client once the new connection is established. These problems are typical of those experienced in a distributed system. The additional expense of implementing a full mobile IP protocol is not thought to outweigh the benefits of full transparency in the case of the NOTUS environment.

NOTUS is currently implemented on top of the Nemesis Distributed Operating System which has been designed to support multimedia applications and provides support for addressable modular units of code. In the test environment, which consists of DEC Alpha workstations on a 10Mbps Ethernet, the cost of an RPC is found to be of the same order of magnitude as a server migration.

3.5.3 Analysis

To the author’s knowledge there has been little work examining mobility support within open distributed platforms. Given the importance of open systems technology,

particularly within industry, this lack of interest is perhaps surprising. The majority of the work carried out in this field attempts to handle mobility as another form of distribution transparency. The author believes that this is the wrong approach. The role of transparency and adaptation in distributed systems platforms is discussed further in section 3.7.

3.6 Other Related Research

This section aims to cover some of the other related research in the field of mobile computing. Much of the work presented here relies on one or more of the concepts or technologies presented in the previous sections.

3.6.1 Wit

The Wit programming paradigm [Watson,94] developed at Washington, requires applications to be custom written to work in a mobile environment. Each application is split into two halves (a process described as *application partitioning*): a mobile portion (typically the user interface) which is resident on the mobile host, and a static portion (often retaining the data parsing functionality) which remains on the fixed network. Each portion contains data objects which may encapsulate multimedia data and program functions. Objects (known as *hyperobjects*) may be migrated, replicated and linked hierarchically in response to resource management decisions.

The mobility of the host is hidden from the application. Indeed, all efficiency and system resource concerns such as caching, prefetching and filtering of hyperobjects are handled transparently by the architecture. To aid Wit in the resource management process, relationships between objects are specified as a graph by each application. Nodes in the graph correspond to data objects, with edges expressing inter-object relationships. Graphs of objects, written in the Tcl scripting language, are communicated with the Wit architecture using an API. This approach contrasts with those typically employed by the mobile file system community, where user access patterns are used to pick out data relationships.

This approach is currently being investigated using a network of PCs and SparcStations attached to a high speed wireless LAN. Evaluation applications including World Wide Web browsers and CAD tools are under development.

3.6.2 Adaptive Wireless Information Systems

The work of Rutgers University has concentrated on the design of an architecture to support wide-scale provision and dissemination of information to large number of

mobile consumers [Imielinski,94b]. Rutgers subscribe to the view that networks are largely fixed with mobile support stations providing connectivity to mobile hosts on the fringes of the network. Each mobile support station will support two basic information delivery paradigms: on-demand and broadcast (publishing mode). On-demand information must be explicitly requested by consumers. Broadcast information is transmitted to all hosts within a cell. Pages of information from a particular service are allocated based on the frequency of request to either of the two "channels". In this manner it is hoped that the most efficient power and packet efficiency can be achieved.

In this model, future information services are expected to be classifiable in terms of geographic scope. For example, at the finest granularity, pico-cells may transmit information about parking space availability, building layout or advertisements in a shopping centre. At the largest granularity, general information is expected such as stock market information or news bulletins.

The published information channel is expected to have a number of advantages. For example, information flow is uni-directional from the MSSs to the clients, which avoids use of the clients' transmitter to save battery life. In addition, protocols can be developed to allow a client to *doze* while waiting for information to be broadcast, which would make further power savings. To facilitate this mode, a network interface which is capable of filtering information based on a particular multicast address is required. Such an interface would only wake the client when information on a particular address was received.

Currently none of the architecture has been implemented, although changes to the world wide web protocol (HTTP) have been proposed.

3.6.3 MobileChannel

MobileChannel [Cho,94] is a tool constructed on top of the ISIS [Birman,89] distributed systems platform which aims to support host migration through group communication. ISIS provides two fundamental functions: group management and group communication. Two communications primitives are supported for group communication: CBCAST and ABCAST. CBCAST implements reliable causal ordering delivery to each group member. ABCAST, implemented on top of CBCAST, gives total ordering semantics, i.e. every group member observes events in the same order. Moreover, these primitives are used to attain *virtual synchrony*, which is the illusion that every event happens synchronously in the system, despite processes receiving event messages at different physical times.

The MobileChannel tool is a lightweight agent which is responsible for converting mobile client messages into ISIS events which may then be transmitted to the host's primary server on the fixed network. The primary server uses ISIS to propagate updates to its replicas using multicast. In MobileChannel, migration is handled as an intentional "primary switch", i.e. the transition between using one primary server to another (this normally only occurs on server failure in a fixed network).

The virtual synchrony model allows handoff (and consequent change of primary server) to occur without the normal handshaking process between the new server, old server and mobile client since state information is already propagated to all possible servers.

The current implementation is on SparcStations with PC mobile clients. A 2Mbps WaveLAN wireless LAN has been used to evaluate the MobileChannel performance. The test environment is not able to locate mobile clients nor are the WaveLAN devices capable of roaming and handoff. Therefore handover has to be software triggered and emulated.

In order to scale the architecture, a hierarchical approach is suggested where the server group membership changes with host location. For example, as the client host migrates, servers near the host join the group replacing more distant hosts in the group. In this fashion the servers surrounding the host (termed a flock) are consistent enough to take over a primary server. The ISIS distributed systems platform and MobileChannel tool were originally conceived at Cornell.

3.6.4 Bay Area Research Wireless Access Network (BARWAN)

The BARWAN project at Berkeley [Katz,96b] aims to develop a scalable architecture to support the integration of wireless access technologies which "overlay" one another to offer seamless connectivity to mobile hosts. Furthermore, a testbed, employing a range of services from indoor infrared and wireless LANs to broadcast satellite services, is being constructed to validate the architecture (shown in figure 3.3).

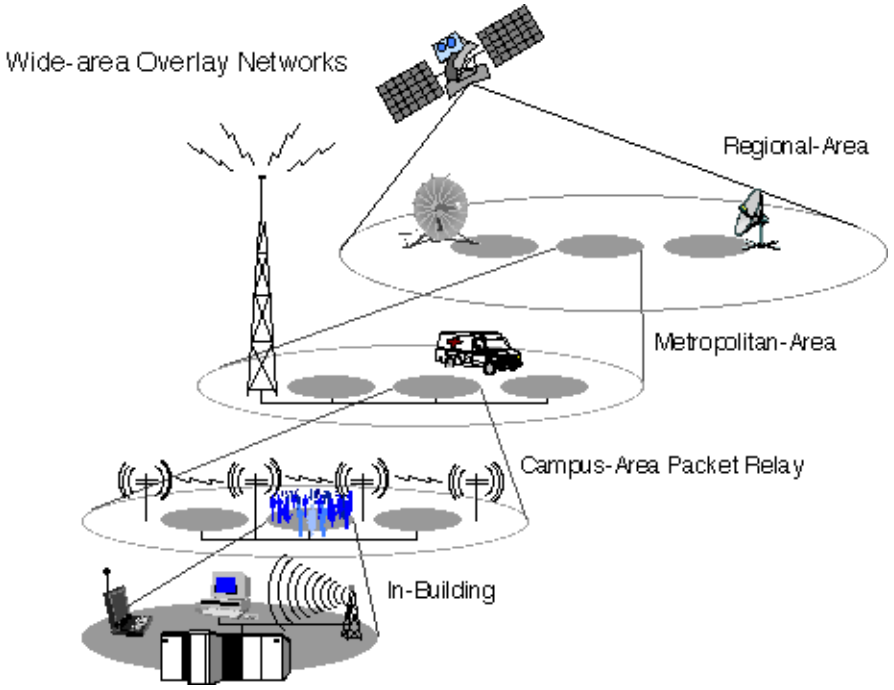

Figure 3.3 - Overlay network testbed

Wireless technologies are seen as either being *black pipes* or *cooperative networks*. A black pipe (most commonly a wide-area subscriber service) provides communications services with little control over routing, priority or network QoS. In contrast, a cooperative network makes low level information available and may support a control interface. The BARWAN architecture is designed to layer over these technologies to provide seamless integration with *vertical handoff* between technologies. For example, if a group of hosts are using an IR LAN within a room and a further user joins them, the architecture may choose to handover some of the connections to an overlapping RF LAN to balance the load.

Support services are provided to enable applications to initiate handover between overlays, enquire about the QoS provided by the network and gracefully adapt to the changing characteristics of the network. More specifically, transmissions from mobile hosts are strongly typed (the type system is dynamically extensible). Thus based on media type and network information, the support services can employ bandwidth minimisation techniques on the applications behalf. For example, bitmaps may be sent in raw, compressed, or highly compressed (lossy) forms depending on link quality. In addition, the project has developed a video transcoder [Amir,95b] which can translate motion-JPEG at 1Mbps to 256Kbps H.261 for wireless transmission at between 21 and 30 frames per second (depending on the quantity of motion in the video stream).

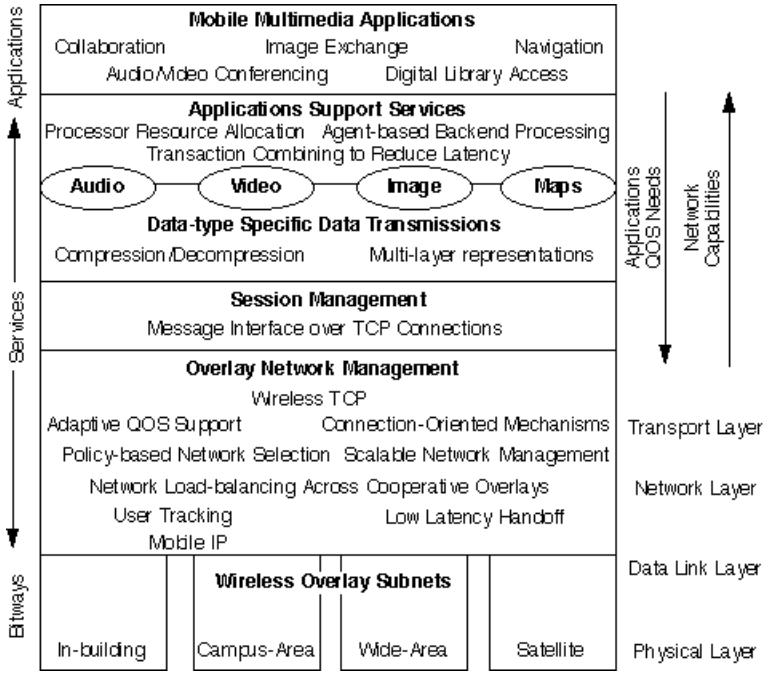

Figure 3.4 - BARWAN layered architecture

Figure 3.4 illustrates the conceptual architecture for the wireless overlay internetworking and mobile applications support services. The architecture consists of a number of layers, the lowest of which is the interface to the overlay sub-networks themselves. Each network type has its own IP address which is managed by a new protocol called Overlay IP. Overlay IP provides the functionality of a conventional mobile IP protocol (see section 3.1) with additional policy driven mechanisms for determining which sub-network to choose for a given packet.

The Overlay Network Management Layer handles the routing of packets across heterogeneous networks. More specifically, this layer is responsible for choosing the most appropriate sub-network to provide end-to-end connectivity given an application's QoS requirements. In addition, low latency vertical handoff support is provided within this layer [Seshan,95]. The handoff algorithm makes use of IP multicast and base station location information to reduce the latency of connection handoff. Essentially, as a host migrates, a multicast group is formed containing the current base station and those that the host is likely to encounter given its current speed and trajectory. Packets destined for the host are multicast to the base station group. Each base station is therefore prepared with the most recent messages for the host when handover occurs. The Overlay Network Management Layer additionally supports the mobile TCP connection-oriented communications service (described in section 3.2.2).

The BARWAN architecture is being designed, deployed and evaluated with the following wireless technologies: a WaveLAN local-area network, a Metricom

wireless packet relay network (both on campus at Berkeley and in the East Bay area), CDPD from GTE Mobilenet and DirectPC Direct Broadcast Satellite services from Hughes. The low latency handoff, reliable wireless transport and video transcoding mechanisms are all implemented. In addition, a low bandwidth web browser has been developed which utilises an image proxy service (to adjust bitmap quality as outlined earlier). User initiated vertical handoff between multiple wireless LANs and the Metricom service is operational. Network initiated handoff between local and wide-area networks is currently under development. One of the first applications which will be developed over the architecture is to provide medical personnel in the Bay Area with a tool for consulting specialists about diagnoses at emergency incidents.

3.7 Adaptation and Transparency

The work presented in this chapter demonstrates the wide ranging impact that wireless communication has had on system and application level software. Note that, a significant number of projects have taken the approach that mobility should be handled transparently. Indeed, it is worth noting that mobility transparency is required to support legacy applications. However, mobility transparency presents a number of problems.

Consider the Walkstation project, first mentioned in section 2.3, an application using a Walkstation interface will not be aware which networking technology is carrying its data. The application has no way of knowing the implications of transferring a large amount of data and, in addition, can perform no optimisation (for example, attempting to save bandwidth or match its packet sizes to those of the network). Furthermore, the application has no control over which network is used for any given interaction, or of specifying how the information should be treated (for example, based on urgency or network QoS requirements). Although representing an extreme case, the argument applies to any scenario where an application is kept uninformed of changes in its supporting infrastructure.

Significantly, a number of independent research efforts (namely MosquitoNet, Odyssey and BARWAN), have each highlighted that even legacy applications will perform poorly in scenarios such as these. Indeed, Baker observed in a recent paper :-

> "*Bandwidth, latency, bit error rates, security, and cost can all differ significantly from one type of network to another. We believe it may be advantageous to inform upper-layer network protocols and some applications of these changes so they can adjust their behaviours accordingly*" *[Baker,96].*

Examining the research presented throughout this chapter: efficient mobile IP, wireless TCP and mobile RPCs provide valuable technologies for mobile environments, if packets are to be efficiently routed to mobile hosts and bandwidth is to be utilised efficiently. Mobile file systems, such as CODA and Jetfile, are essential for supporting legacy applications such as document editing or software development. However, these protocols and services will be required to provide information to higher layers, if applications are to adapt.

Therefore, with this additional information, services and applications can apply any of the vast array of techniques to adjust their communications requirements to the available resources. For example, data fidelity or CPU time may be traded against bandwidth by using media specific techniques such as those in the Odyssey Wardens, mobile DCE Resource Manager or the BARWAN Application Support Services. Intelligent filtering techniques or agent based technologies such as TACOMA or Ara may help to reduce bandwidth requirements. Services may be migrated or alternatives found as with Rover, Traded handoffs and Mobile DCE. Furthermore, the application, either itself or by providing feedback to the user, might avoid many communications related problems simply by changing the expectations of the system (for example, postponing a particular task until later).

The need for application level adaptation is intensified by the introduction of heterogeneous networked environments (proposed in chapter 2 and constituting the BARWAN testbed) and the demands of advanced applications. As highlighted in chapter 1, experience suggests that suitable distributed systems platforms can greatly simplify the development of advanced applications. Furthermore, environments comprising heterogeneous integrated architectures will require open distributed processing platforms. Accepting that this is the case, it is clear that these platforms must also provide facilities to enable application level adaptation.

Currently, only the Mobile DCE platform attempts to handle mobility within a heterogeneous network. Mobile DCE classifies the use of resources and describes the functionality of applications using finite automata. Thus, it is suggested that by providing generic bandwidth minimisation mechanisms and high level abstractions of the network and domain services, host mobility may be handled transparently using a set of simple selection policies. Moreover, more complex policy decisions may be handled by the bandwidth and cost management subsystem and through an API enabling the application to select from the supported mechanisms.

In the author’s opinion, it is both burdensome on the application developer and not necessarily practicable to classify all resources and applications in this way. Furthermore, it is unlikely that generic components are able to supply a broad enough

range of translation and decision making mechanisms to cope with the requirements of advanced applications in heterogeneous networked environments. A more realisable approach would offer a streamlined platform which provides a framework for managing the network and application QoS. Then, through a process of feedback to and control by the application, enable it to adapt by applying task and data specific knowledge offset against the currently available resources.

3.8 Summary

This chapter has presented a comprehensive survey of system services and application level architectures to support mobile computing. Current research is largely providing abstraction over changes in the supporting infrastructure and yielding complete transparency to user applications.

Based on these surveys it has been argued that transparency does not allow applications to make the best use of the available communications link. Furthermore, applications will require information about changes in the network QoS to be made available so they may respond by adapting their behaviour.

The remainder of this thesis focuses on the development of an open distributed platform which provides a QoS based architecture for monitoring and feedback of network QoS to support adaptive mobile applications in heterogeneous networked environments.

Chapter 4

Advanced Mobile Applications in the Utilities Industries

4.1 Introduction

This chapter introduces some background information that is necessary to aid understanding of the application and platform presented in the next two chapters. The work was carried out within the context of the MOST project [MOST,92]. The project aimed to develop an advanced mobile application to support field workers in the electricity industry (although the concepts demonstrated are intended to be equally applicable to other complex distributed utilities such as gas or water). As part of this work, an extensive requirements capture exercise was undertaken which had a direct influence on the design of the application and supporting platform.

This chapter describes the results from this study. More specifically, the structure and function of a typical electricity utility company is discussed. An example scenario is presented which demonstrates a number of shortcomings in the current working practices of the industry and illustrates how a collaborative application could assist in overcoming these problems. Lastly, a set of requirements for the application detailed in the next chapter are highlighted.

4.2 The Power Distribution Industry

The U.K power industry is divided up into fourteen divisions, each on average responsible for 16,000 square kilometres of geographical area. Each division must provide electricity to approximately 1.8 million customers, which requires around 54,000 kilometres of distribution network. The following sections describe how this vast power distribution infrastructure is organised and maintained.

4.2.1 The Power Distribution Networks

Each division's network is structured hierarchically into a number of constituent sub-networks each operating at a different voltage, typically 132kV, 33kV and 11kV (see figure 4.1). These sub-networks are referred to as high-voltage or HV networks. Below 11kV, the network is considered low-voltage or LV.

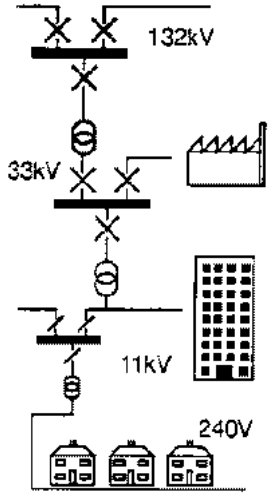

Figure 4.1 - Power distribution network operating voltages

The voltage level and network density/complexity are inversely proportional. The national grid (which supplies each division) operates at the highest voltages, but additionally covers the greatest area and has the fewest interconnections. At the other end of the scale, the 11kV network supplies substations within residential and commercial developments. Consequently, the 11kV network spans less area but is far more densely interconnected. Below 11kV, the LV network is fragmented into 4kV, 480V and 240V for supply to individual customers (the higher voltages being for factories and light industrial complexes).

The 132kV network is made up of a small number of substations interconnected by multiple supply paths to allow the maximum configurability. The network is dynamically configured to cope with fluctuations in demand and any emergency supply interruptions that occur. The 132kV network is simple enough to be

represented on an electronic wall chart which, in addition, monitors and controls remote actuators. The actuators supply voltage and current telemetry to the control centre and enable switching of power between substations. Generally, it is not cost effective to deploy remote sensors and actuators in the lower voltage HV and LV networks.

The 33kV network consists of many more substations than at the 132kV level. However, the number of interconnections between these substations is considerably reduced: a typical substation has two incoming *feeders* which are in turn routed to a number of smaller substations which transform the voltage to 11kV. The 33kV network is too complex for conventional electronic representation and so is typically represented diagrammatically on a series of wall charts. Pins or magnetic markers are used to indicate the state of switches within the network. Several systems are under development which aim to represent these complex charts on a computer without losing the meta-information that experienced control room engineers draw from the conventional wall mounted diagrams.

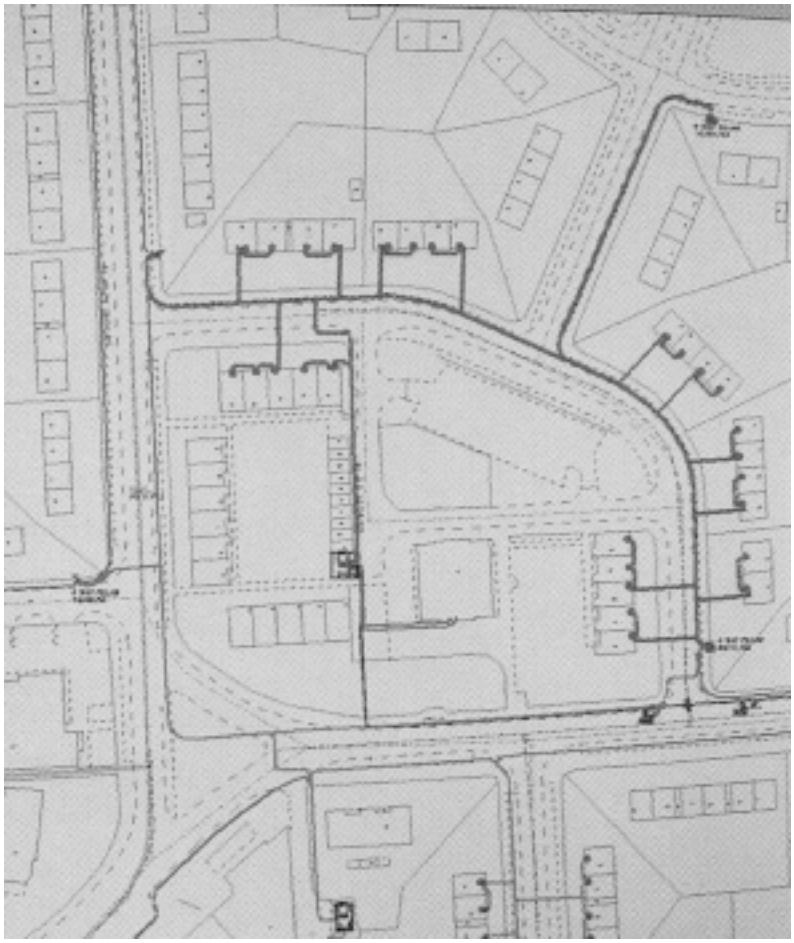

Figure 4.2 - Layout of LV power supply

The 11kV network is the voltage which supplies the substations in individual residential and commercial areas (see figure 4.2). Generally, the type of power transmission technology corresponds to the requirements of the geographical area they supply. For example, in rural areas overhead lines are most commonly used, whereas in towns and cities cables are frequently buried underground. In both cases, the cables are laid out to mirror the layout of roads and houses to ease location and maintenance. Below 11kV, the LV network is too large and complex to be laid out on a wall chart. Instead, the network is represented on a large number of individual sheets which are kept and updated at a central location.

4.2.2 Operational Issues

Repair work on the power distribution network is largely event driven. The work events are scheduled according to some measure of urgency: external (fault driven) events are considered more significant than internal events. External events are scheduled "on-demand" in response to power outages. The urgency of an external event is driven by how quickly power can be restored. Thus, if power can be restored to the affected area by some alternate supply, the remainder of the repair work on the faulty section will become an internal event (unless the repair is trivial). Internal events are scheduled in advance according to criticality. For example, repairing a redundant feed to a substation in case the primary feed fails would be considered more urgent than the ceaseless programme of cable renovation (in a large division, 50 year old wiring that is still in service is not uncommon).

The system which controls the work activity can be grouped into the three domains shown in figure 4.3 (some overlap of tasks is inevitable in the real system).

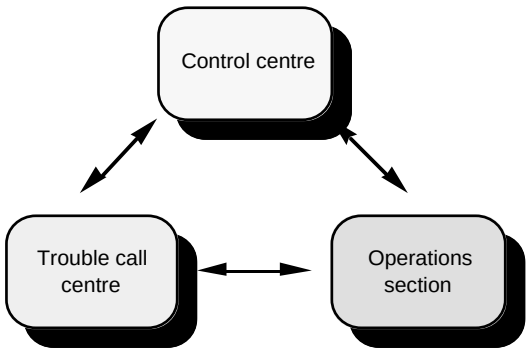

Figure 4.3 - Components of the work scheduling and maintenance system

The trouble call centre is the utility's interface to the general public. Significantly, the centre is also the primary mechanism through which faults in the distribution network are discovered. In the event of an abnormality or malfunction, telephone calls from people suffering from *outages* (power cuts) allow the company to gauge the

location and the size of the fault, i.e. calls coming in over a widely dispersed geographical area would signal a fault at a higher voltage level than calls from customers located along a single road. The trouble call system incorporates a voice processor to enable the automatic filtering of reports relating to faults which are already under repair. In many instances the outage is caused by third party damage such as the laying of cable television networks. In these scenarios, the third party contractor may report the damage directly, pinpointing the fault without the need for detailed diagnosis.

The control centre is responsible for coordinating all operations which affect the state of the network in order to preserve both integrity and safety. Their remit includes :-

i) Managing disjoint teams of engineers that are physically distributed in largely fixed but separate locations,

ii) coordinating and maintaining the view of current network state, and

iii) talking the teams sequentially through operations on the network.

The network state is maintained on a number of paper or electronic wall charts (many utilities are digitising their network diagrams for inclusion in a computerised representation system). In addition, the control centre gathers tele-control information from sensors in the HV network which can sometimes highlight faults. For example, if a primary substation detects a sudden drop of 100 Amps which corresponds to the load of a particular sub-network then it is likely that the lower level substation has tripped on detection of a local fault.

The operations centre schedules work for the maintenance engineers as well as serving as a base for the field staff and a reference library containing permanent copies of the division's maps and schematic records. When the trouble call centre reports a new fault, the operations centre is responsible for allocating manpower to carry out the first line investigation. If the fault is on the LV network then, in a high percentage of cases, all that may be required is for an emergency engineer to replace the fuse at the substation. If replacing the fuse does not fix the problem, then the emergency engineer will report back to the operations centre to call out a fault engineer to diagnose the problem. If the fault is on the HV network then a fault engineer will be required with appropriate information on the geography and network configuration. Once the fault has been diagnosed, any switching of the HV network must be done under consultation with the control centre in order to maintain network integrity.

If power restoration can be fully achieved by reconfiguration, then emergency switching restores the supply and the repair becomes a scheduled work item. Only where full supply is not possible will an emergency work item be scheduled. An emergency work item may take staff from planned work depending on a balance of who may respond the most rapidly against the urgency of the fault and the deferability of the work already undertaken by that person. Such a balance requires a high degree of accountability and coordination between members of staff at the operations centre and the field workers themselves. Should a field engineer be transferred to a different item of work then they will require access to the relevant maps and schematics for that area: either an engineer must be limited to the area that he can carry information on, or must detour via the operations centre to collect new information.

The field workers themselves are organised into a number of specialisations depending on experience, expertise and seniority. There is a requirement for cooperation between field workers, particularly where some have specialist knowledge or experience which is needed before making decisions. For example, portions of the LV network can be ‘checked out’ to fault engineers who will supervise all the repair operations on that section of network until it is ready to be ‘checked in’ (in each case by talking through a sequence of switching steps with the control centre). This deferring of responsibility helps amortise the cost of a central point of control by using experienced field staff. However, subordinate field workers will often need to contact more experienced field staff regardless of where they might be located. In addition, further maintenance to the section of network administered by a particular field engineer will call for a degree of collaboration before work can begin.

Once the engineers have completed the repair work, the fault is documented by annotating the *composite mains record* (CMR) for that region to indicate switching changes or new wiring (the CMR details the geographical layout of the region, overlaid with LV schematic information). The margins of the CMR are often used for comments such as the particular grade of cable used or to describe the depth and location of the repair in relation to another cable or feature. This highlighted record is submitted to the operations centre who disseminate it to the draftsmen for inclusion in the permanent record, and to the control centre to update the picture of the network state.

Typically, field engineers will experience different latencies on updates sent to the control centre depending on the nature of the repair: HV network changes will be updated quickly (typically within 24 hours), whereas LV changes could take a week or more to be accurately reflected. To operate in such an environment there is a requirement for rigorous safety procedures to allow an engineer to discover

inconsistencies between the perceived network state at the control centre and the real state at a point in time.

4.2.3 Physical Security

Work on a electricity distribution network is by necessity safety critical. In the absence of technological solutions, strict physical mechanisms have been developed to enforce safe operation. To give some examples, when field engineers take responsibility for a section of LV network they are required to physically travel to the operations centre and obtain the *linen* for a particular region before they can negotiate for control with the control centre. The linen is literally a piece of linen cloth with a copy of the region's CMR. Any other engineer who is interested in that particular section of network must either negotiate with the engineer possessing the linen or, more frequently, wait for the engineer to finish with the section.

A further example is the pervasive use of physical locks and keys, particularly within substations. Each fault engineer will carry a number of unique padlocks and keys which can be locked on to physical switches to prevent any other engineer changing the state of the switch (activating or earthing for example) without prior contact with the engineer. When the fault engineer has to coordinate teams of people on the same section of network, a *lockout box* is often used. The lockout box is literally a metal box with four locks, three with unique keys and a fourth with a common key that is allocated to all fault engineers (therefore a fault engineer has to be present to open the box which preserves the chain of authority). Before work can begin on the section of network it must first be isolated then earthed by throwing switches at the substations at either end of the section. To ensure the section remains isolated, the switches are padlocked into position and the keys locked in the lockout box (to cope with larger groups, lockout boxes may be 'chained together' by placing one of the unique keys in another lockout box). Thereby only with the consent of all key holders can the section of network be re-energised which ensures the safety of the workmen.

4.3 Information Technology Support

The following sections examine the current role of information technology within the electricity utilities. More specifically, the communications infrastructure and a range of supporting computer systems are described. In addition, a number of limitations with the current system are highlighted.

4.3.1 Communications Infrastructure

Most of the utilities companies in the U.K operate their own analogue PMR systems (section 2.2.1.2 described PMR systems in more detail) within a dedicated band of frequencies. The PMR systems are based on the MPT 1327 trunking standard which allows analogue point-to-point voice calls to be established and limited status code messages (numbers 0-31) to be sent using the control channel. The status codes are used extensively to operate a call-back queuing system at the control centre. The system is half-duplex and makes use of vehicle mounted transceiver units with push-to-talk hand held microphones.

Anecdotal evidence gathered during the requirements capture suggests three primary criticisms of the current system: coverage, lack of prioritisation and lack of a group channel. These issues are considered in more detail below.

Common channel

The current PMR replaced an earlier system of handheld push-to-talk walkie-talkie style radios. Although the PMR systems are far more sophisticated and offer greater capacity and coverage than the previous system, the connection-oriented communication style has changed the way engineers work. With the original system all traffic was shared with any receivers that were in range. This shared channel had two advantages: firstly, in a real emergency, engineers could hear which messages were truly urgent (for example, an injury) and secondly, by hearing all the communication going on around them, engineers were able to implicitly balance the load of repairs, knowing where colleagues requiring help were located and whether or not they could assist. The MPT 1327 standard does allow for group calls to be established. However, the group establishment procedure is too complicated for everyday use and requires all the parties in the group to be known beforehand (new members cannot be added dynamically to an ongoing call).

Traffic prioritisation

The PMR systems used in the divisions are heavily over subscribed, particularly when engineers have to access the control centre (which is often seen as a bottleneck). Engineers commonly wait for over half an hour before the control centre is able to call them back. Furthermore, the calls are not priority based, therefore an urgent or significant communiqué has to wait on the call queue as long as any other message.

Coverage

The coverage of the system can be very poor; engineers can experience coverage blackspots where they must drive for up to thirty minutes before regaining sufficient coverage to contact colleagues. Indeed, engineers have begun to unofficially carry cellular telephones just for cases such as this. The PMR system represents such a huge investment to the utilities companies that it is difficult to justify the expense of improvements to increase the quality or coverage of the system: the current PMR systems have not been upgraded to use more recent technology. Furthermore, under cases of extreme stress such as emergency situations caused by severe electrical storms, the PMR system can suffer from both insufficient capacity and interference problems from the storm.

There are initiatives within most utility companies to move to newer technology. The most likely candidate is digital PMR technology based on the TETRA standard, although some are conducting trials using public telephone systems such as GSM to examine their acceptability and cost effectiveness.

4.3.2 Computer Systems Support

Historically, the utilities industries have made extensive use of computer technology throughout their companies. A significant number of independent systems have become established for dealing with particular facets of the company's interests.

System	Purpose	Location	Hardware
Capital investment management	Costing extensions and repairs to the distribution network	Centralised	Mainframe
Project records management	Recording and monitoring work project progress	Centralised	Mainframe
Trouble call	Collates faults and customer calls	Centralised	Mainframe
Plant records and maintenance	Tracking the quantities and location of plant	Centralised and depots	Mainframe
Tele-control load data analysis	Network load analysis and prediction suite	Centralised	Mainframe
Work instruction management	Job dispatching and completion records	Centralised	Mainframe
Geographical information (GIS)	Recording digitised schematic and geographic records	Centralised	Networked workstations
Switching schedule production assistant	Expert system for switching schedule production	Centralised	Mainframe or stand-alone workstation
Network representation	Replacement for electronic wall charts	Control centre	Workstations
Data in the field	Trial of CD-ROM based CMR records in the field	Mobile	Portable computers

Table 4.1 - Overview of divisional computer resources

Table 4.1 illustrates the wide range of systems that are being used within a typical utility company. These systems have been developed over many years, largely as a

result of periodic and massive infrastructure investment. The value of the systems, both financially and in terms of the information now contained within them, prohibits them from being replaced. Hence, the typical utility will have a range of legacy systems which must be integrated with newer systems in order to be viewed as a cohesive unified resource. As a direct result of this incremental development, these systems are based on a wide range of hardware and software technologies. In particular, the mainframe systems are based on wholly proprietary technologies which makes interworking particularly difficult (and hence many utilities are interested in open systems technologies).

In addition to internal system integration, there is often a need to work collaboratively with adjacent divisions. Currently, the utilities industry supports a limited form of inter-divisional cooperation through the adoption of a common PMR system. Subject to authorisation, a field engineer can work under the domain of an adjacent division to conduct 'out-of-area' work. While working in the foreign domain, the engineer has access to his home domain through the PMR system. Such a facility is often essential for dealing with HV faults with the national grid since switching operations may affect multiple divisions and thus require coordination. It is anticipated that, in addition to providing support for communication while in the foreign domain, future systems will ensure out-of-area engineers can access their home data services also.

The recent establishment of the *common street-works register* (CSWR) has led to an external requirement for the electricity utilities to interwork with other utilities such as gas or water. The role of the CSWR is two fold: firstly, it will enable utilities to be aware of each others' infrastructure to reduce the amount of third party damage and, secondly, it will permit the utilities to loosely cooperate so that jobs on a common area of network are scheduled concurrently. In order to support the CSWR the companies are initially only required to be able to exchange textual messages relating to scheduled work. However, it is anticipated that more complex interchanges will arise as fully computerised public views of the distribution networks are developed. Due to the mix of deployed technologies and the separation of administrative control over each division's computer systems, it seems clear that open systems technology would be able to assist in the foundation of the CSWR.

4.4 Need for Collaboration

This section discusses how the communication and computer support aspects already outlined are used, based on information gathered during the requirements capture phase of the MOST project. More specifically, a hypothetical scenario is

presented which highlights the collaborative nature of many distribution network repairs.

4.4.1 Requirements Capture Process

To gain insight into the working practices of a typical division, the project team conducted a series of interviews as part of the requirements capture process. The team interviewed seven field engineers with a range of responsibilities and experience chosen from a specific U.K. division†. More specifically, the interviewees were responsible for the following jobs :-

- new business engineer,
- general construction engineer,
- under eaves wiring maintenance,
- fault engineer,
- trouble call,
- operations centre engineer, and
- overhead line engineer.

These interviews afforded the project with invaluable background information concerning the structure and operation of a typical utility and, significantly, identified a range of limitations with their current working practices. Based on this information, a number of example distribution fault and repair scenarios were identified.

The following scenario was selected as being representative of the studied working practices. The scenario is used in this thesis to highlight specific problems experienced by field engineers (and also illustrates the uses and interaction of the various IT components).

† A non-disclosure agreement covering all work conducted under the MOST project ensures that the company must remain anonymous.

4.4.2 A Generic Scenario

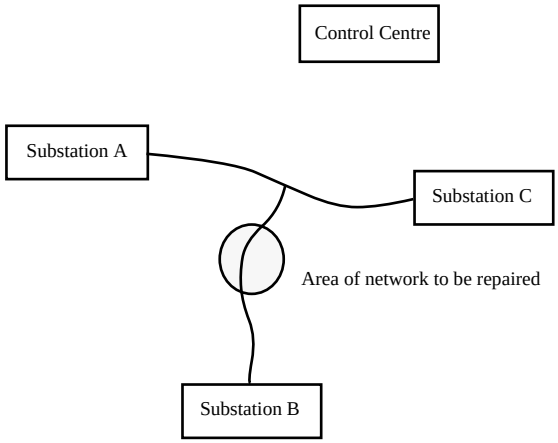

Figure 4.4 - Maintenance of the power distribution network

Figure 4.4 represents a section of the LV power distribution network. The figure shows the control centre and three substations, labelled A, B and C. The fault, indicated by the shaded circle, can be considered to be on the HV feed to substation B. It is most likely that a fault such as this would be detected as a result of customers whose supply is fed from substation B calling in to report outages. The trouble call system would log the reports and highlight the fault on a list of pending jobs. An emergency engineer will pick up the task from the pending list upon returning to the operations centre.

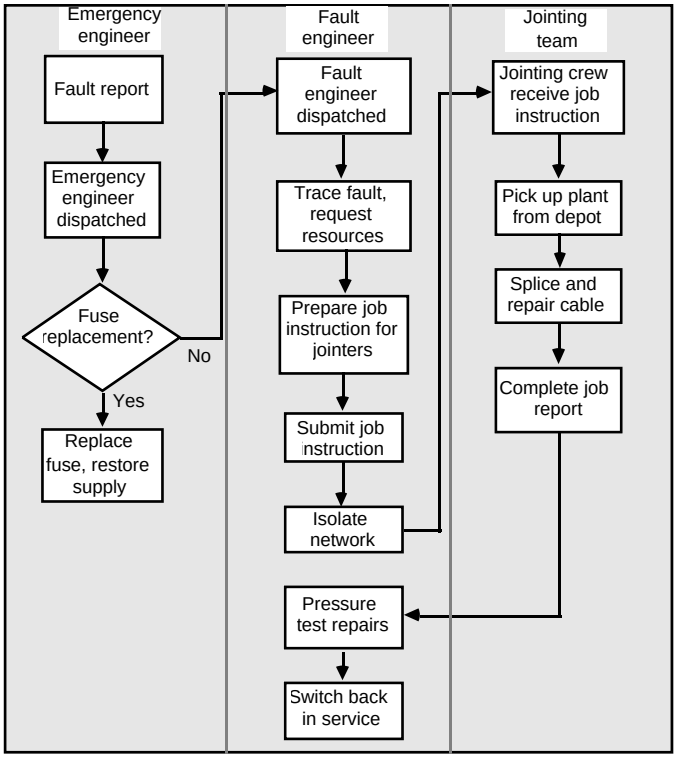

Figure 4.5 - Basic fault finding and repair procedure

The basic sequence of steps to effect a repair is outlined in figure 4.5. The emergency engineer would analyse the fault report generated by the trouble call centre and, from the list of attached customer reports, estimate the most likely point for the fault to occur (substation B). At substation B, the engineer examines the state of the trip switches and fuses in the substation to determine which part of the LV network contains the fault. In this scenario, none of the switches would have activated and the engineer could deduce that the fault was on the feed to the substation itself. The emergency engineer is responsible for restoring supply to the affected customers as quickly as possible. If supply cannot be restored simply by replacing a blown fuse, the engineer must request a fault engineer to conduct a longer term diagnosis. The emergency engineer must return back the operations centre to schedule the work using both the project records and work instruction management systems.

The fault engineer would begin investigation at substation B, based on the emergency engineer's report, then work up the supply hierarchy until a substation was found that had detected the fault (in this case the engineer would examine substations A and C). The engineer would require the CMR for the affected region and would consult the 11kV wall chart at the operations centre to obtain an overview of the connectivity of substation B. The search area would be narrowed down to the section of cable between the pair of substations A and C and the junction supplying the primary feed to substation B.

Once the fault has been specifically located and analysed, the engineer can judge whether supply can be restored temporarily to the affected customers from another substation or whether the repair constitutes an emergency work item. If supply can be restored, the engineer will consult the control centre using the PMR and, if permissible, conduct the emergency switching. The remainder of the repair work will then become a scheduled work item. Upon returning to the operations centre, the fault engineer would need to update the wall chart to reflect the new state of the network for other field engineers.

Item	Location	Plant Item	Operation	Operator	Time
1	Cross	8288/1/2 OS	Check open		
2	" "	" "	Close to ON		
3	Cross	Reifield OS	Open to OFF		
4	" "	" "	ASL & PCN		
5	Reifield	Cross OS	Open to OFF		
6	" "	" "	ASL & PCN		
7	" "	" "	Prove Dead		
8	Reifield	Cross OS	Close to CIRCUIT earth CME 1		
9	Cross	Reifield OS	Close to CIRCUIT earth CME 2		
10	Hannard Down	FUSE	Identify point of work		
11	" "	" "	Prove Dead		
12	" "	" "	IssuePTW...Replace insulators		

Figure 4.6 - A sample switching schedule

Before further work can be carried out on the fault (whether as an emergency or scheduled item) the fault engineer must submit a switching schedule (see figure 4.6)

to the control centre using the internal mail system. The schedule describes in detail the stages involved in carrying out the work and, in particular, the sequence of switching which must be carried out to ensure that the work can be conducted safely (i.e. the section of network being operated on is isolated and earthed) and with the minimum of disruption to customer supply. The control centre checks the switching schedule against the electronic wall chart illustrating the network state (this may be held on its computer system) and approves or rejects the schedule accordingly. Expert systems may be used by both the control centre and the field engineer in the development of the switching schedule [Cross,93] although at present these systems are not integrated with the centre's representation of the global network state.

The fault engineer will use the plant records and maintenance system (either from the operations centre or nearest depot) to requisition and locate plant for completing the repair. When the repair work is to be carried out, additional field engineers will be required at the appropriate switching points (substations A, B and C). The control centre then acts as a central coordinator, using the PMR system to instigate sequences of switching at each substation. The central coordinator serialises the switching activities of the engineers to maintain the correct ordering to ensure the work is conducted safely. In addition, using the PMR enables the engineers to reconcile any differences between the notion of state at the control centre and the actual network state. Thus, during the process the centre will gain a more up-to-date picture of the current state of the network.

Once the switching has been completed, the section of network will be isolated and repairs can begin. The repairs themselves will require the fault engineer to collaborate with a jointing crew to dig up and patch the damaged section of cable. The work instruction management system will be used to issue the jointing crew with the necessary work instructions and, additionally, will allow the fault engineer to find out when the work has been completed. Before the repaired section of network can be switched back in to service, the fault engineer must 'pressure test' the repair. Pressure testing ensures that the repaired section will handle the necessary load and, in addition, that no other faults are present within the repaired section that were masked by the primary fault.

When the repair has successfully passed inspection, the control centre is again involved to return the network to its nominal operating state (the process is very similar to the switching out process). When the entire repair has been completed the engineer must return to the office to complete the necessary administration using the project records management system.

4.4.3 Analysis

There are clearly a number of disadvantages with the current system which impair the efficiency and scalability of the architecture. Some of the more significant restrictions placed on the system are outlined below :-

Available network state

The long latencies in updating network state information forces field workers to adopt a wide variety of physical safety mechanisms to ensure their own safety (a selection of these are discussed in section 4.2.3). In addition, all network state modifications require the direct intervention of the central point of control. Field engineers often experience long delays while in contention for access to the control centre (which would increase if the network were to be expanded).

Serialisation

Serialising all collaborative activity through the control centre rigorously enforces safety. However, once again, forcing each engineer to repeatedly establish contact with the central point of control greatly impairs the pace of the collaboration.

Centralised computational resources

Field engineers require access to a wide variety of systems in order to obtain information and requisition resources for working on the power distribution network. The vast majority of systems are only available in the operations centre which imposes a significant number of time consuming journeys, again reducing efficiency.

Locality information

The open channel PMR system first used by field engineers enabled them to assist each other and balance repair load implicitly. For instance, a field engineer requiring a jointing team would know the location of the nearest team and allocate the task to them. In addition, an engineer requiring temporary assistance or a piece of common plant equipment would not be able to borrow from another engineer; instead, the resource would have to be requisitioned. The current system does not allow an engineer to know the location of their colleagues.

The problems with the control centre are further exacerbated during multiple emergency situations where a number of unscheduled work items are generated (such as the outages caused by a major electrical storm). In such a situation the control centre tends to become inundated by messages from field engineers requesting

information about, or permission to start work on, various parts of the electricity distribution network.

A number of these disadvantages could be overcome by providing field engineers with a computer based support tool to promote peer-to-peer collaboration. More specifically :-

- Network state information could be checked out to mobile hosts to reduce the load on the control centre and improve the availability of up-to-date information (and thus promote safety). The engineers could collaborate without serialisation through the control centre and check in the repaired network upon completion (reducing the update latency).
- Collaborative information sharing tools would enable engineers to coordinate their activities with co-workers and enhance conversational dialogues with multimedia information such as diagrams and annotations, thus reducing some of the ambiguity inherent in a voice only collaboration. For example, an engineer could highlight the particular switch they are about to activate. Furthermore, such a system would simplify the discussion of alternative strategies and enable consultation with remote expertise.
- Providing remote access to operations centre systems such as project record management, plant records and maintenance and work instruction management, would enable engineers to drastically reduce the number of times they have to return to the operations centre to access resources and coordinate with colleagues. In addition, on-line information such as safety reference manuals and CMR records would minimise the amount of travelling still further.
- Semi-automatic job dispatching and report completion at the location of the repair would reduce the time taken to initiate and document changes within the network and additionally reduce inaccuracies sometimes introduced during the transcription process.

The benefits outlined above can be broken down in to a series of requirements which must be met by a system aiming to provide field workers with information technology support. The next section considers these requirements in more detail.

4.5 Requirements

Based on the information gathered during the interviews, meetings and workshops which formed the requirements capture process, the following requirements were identified for a system aiming to support field workers in the utilities :-

Mobility

Field engineers are inherently mobile, so the application must run in a mobile environment. However, most engineers travel in cars and vans to enable them to carry sufficient information, tools and equipment to effect a repair. Therefore, the hardware platform does not have to be hand portable.

Multimedia

Field engineers manipulate information in a wide variety of media types including scanned bitmap (or raster) images, line (particularly hand drawn) vector images and text. In addition, support for voice quality audio is essential for effective collaboration as data only information exchange is too restrictive given current safety practices.

Spatially referenced data

Spatially referenced information pervades almost all activities within the utilities. Supporting tools must provide mechanisms for accessing and browsing maps, schematic diagrams and engineering drawings within specified coordinates. In addition, this information must be individually addressable and versioned to ensure that up-to-date information is used as the basis for interactions and network state updates.

Synchronous collaboration

The application should support peer-to-peer interaction of real time multimedia information. The tools would allow field engineers who are not physically co-located to interact using voice and a shared set of maps and schematics which they can view and manipulate using a number of highlighting tools. For example, engineers might use the tools to coordinate a shared switching task or consult someone with specific expertise.

Asynchronous collaboration

In addition to synchronous collaboration, the application should allow looser (asynchronous) collaboration to permit information updates to propagate in a non-time critical way. For example, a store and forward (E-mail style) mode of working would allow field engineers to create messages requesting the current state of a network subsection and have the system forward the message to the control centre at an appropriate time. In a multimedia system, the reply might include an image showing the current state of the network subsection and an

accompanying voice message. This reply could then be processed at a time convenient to the engineer. By removing the synchronisation at the control centre and decentralising control into the field, the control bottleneck would be reduced and the architecture made more scalable.

Interoperability

The application must allow access to centralised resources of one or more utilities (for example, through the CSWR). In addition, collaboration with co-workers must be offered, regardless of an engineer's physical location and the type of system they are currently using. The operations centre operates a heterogeneous hardware and software infrastructure consisting of a variety of large networked mainframes running a number of operating systems including UNIX and VMS. A field engineer, on the other hand, would use a small portable machine running Windows (for example) and need to use a PMR or mobile telephone to dial-in to the fixed infrastructure.

As previously mentioned, the requirements identified in this section form the basis for an application developed to demonstrate the feasibility of providing collaborative tools to field workers over a heterogeneous networked infrastructure. The requirements presented here are brought out in more detail throughout the analysis of the design and implementation of the application prototype in the next chapter.

There are a number of additional requirements which are particularly important in the development of a real system for deployment in the utilities industry. However, these are not addressed in the remainder of this thesis. The most significant of these is safety criticality: field engineers operate in an environment where incorrect or out of date information could cause fatalities. Thus, if the real time network state information were to be distributed into the field, the application would need to support rigorous safety mechanisms to ensure reliable delivery of information and provide guarantees that the state being viewed is either consistent or inconsistent (and make the difference between the two clearly visible). Furthermore, all the voice and data operations on each host would need to be logged; if an injury occurred, the health and safety inspectorate would require a complete transcript of the events leading up to the accident.

Additional requirements not addressed by this thesis include: ruggedising the hardware technology to cope with weather and the wear and tear of everyday use, employing encryption and authentication techniques to avoid sensitive information such as customer records from being intercepted (either by scanning the

communication or physically stealing the computers), and providing trusted access to centralised systems to prevent malicious intervention.

4.6 Summary

This chapter has provided a variety of background information which is an essential basis for discussing the application in the next chapter. More specifically, the chapter introduces the hierarchical nature of the electricity distribution network and discusses how this network is operated and maintained. The current information technology support for field workers within the electricity industry has been outlined. A scenario has been developed which illustrates how a fault is diagnosed and repaired using the existing information technology systems and current working practices. A number of shortcomings have been highlighted.

A set of requirements are outlined which specify an advanced mobile application that addresses the shortcomings in current working practices. A prototype of this application was developed both to address these requirements and act as an experimental testbed for developing a supporting platform. The prototype application is examined in some detail in chapter 5.

Chapter 5

Application Support for Field Workers

5.1 Introduction

This chapter details the design and implementation of an advanced mobile application aimed at providing the necessary IT support to assist field workers in the utilities industries. In research terms, the application had two purposes: firstly, it acted as a testbed for establishing the support requirements of mobile applications and, secondly, it provided a basis for evaluating the supporting platform (described in more detail in chapter 6). Chapter 4 identified a set of requirements that must be met by the application, these are outlined below :-

- mobility,
- multimedia,
- spatially referenced information,
- synchronous and asynchronous collaboration, and
- interoperability within heterogeneous networks and end-systems.

In project terms, the resulting application prototype acted as a proof of concept demonstrator which facilitated discussion during the ongoing requirements capture process. As a result, the design and evolution of the application was influenced significantly by the wishes of real utility companies. The demonstrations culminated in a limited field trial. The field trial consisted of a sanctioned emergency scenario

that was enacted outdoors in which the prototype application was used to assist field engineers in finding a solution.

The project partners hoped that the prototype would act as a catalyst to enable changes in the current working practices of the field engineers that will allow them to work more effectively. This was indeed proved to be the case and EA Technology Limited are currently developing a reduced functionality product based on the concepts trialled in the prototype.

The remainder of this chapter is broken down into three major areas. Firstly, the design of application and the group conferencing architecture is discussed. Next, the main functional components of the application are highlighted. Lastly, a number of implementation constraints and choices are presented.

5.2 Application Architecture

The application prototype was engineered in accordance with a set of fundamental design decisions which guided the development process. This section commences with a discussion of these decisions before proceeding with an in-depth analysis of application functionality itself.

It was decided that each instance of the application should represent a single user or field worker. Therefore, the one application provides a common interface through which that user interacts and communicates with other users. For convenience it is implied that each engineer (application instance) is run on a dedicated portable computer which can be customised to suit the user's individual job structure and personal preferences. Each engineer's portable computer is able to make (intermittent) use of a range of networking technologies including, wide-area connection-oriented technologies such as a PMR or cellular telephone.

The application has been designed to avoid reliance on central services for assisting in the management of group or application state. The nature of the communications infrastructure and the mobility of the field engineers means that intermittent connectivity between the hosts is inevitable. Providing the application as a distributed set of components, one per host, decentralises the conference state to maximise fault tolerance. Furthermore, reliability is increased by using distributed algorithms to maintain inter-application interactions avoiding a central point of failure (a central point might be unavailable over such a network, despite other hosts remaining available). Note however, that in some PMR systems software can be run on base stations providing a convenient point for establishing group state. The

adoption of such an approach would impair the portability of the application to other technologies.

The application is required to perform the following functions for the engineer :-

- Allow the retrieval and display of maps and schematics relating to the power network.
- Establish and maintain multimedia conferences between field engineers (communication of voice plus data).
- Enable engineers to prepare information both off-line (stand-alone mode) or interactively through collaborative discussion within the conference.
- Provide auxiliary tools which allow fewer synchronous interactions, such as E-mail, queries to remote databases, and automated job dispatching and report completion.
- Supply tools for accessing and interacting with remote legacy systems such as project records, work instruction management and plant records and maintenance.

The requirement to support such a diverse range of functions, together with easily predicted impediments of the target platform, such as small physical screen size and the difficulty of using a keyboard in the field, led to the design of the prototype as a "tool-box" of component *modules*. In particular, the user can activate any subset of the available application modules to tailor their application to the task in hand. The architecture is extensible, allowing new modules and functionality to be added trivially (significant as the MOST project requirements analysis process ran for the duration of the project).

Each component module has an independent graphical interface which may be individually rearranged, iconised and sized to enable the current working set of modules to best fit the limited display size. A virtual window manager is used on each portable to enable a larger effective display area. The interfaces themselves make extensive use of colour icons to reduce the physical size of the interface without an attendant loss of information.

At the time the prototype was written there had been no work exploring collaborative interface design for mobile environments. The interface was developed by extrapolating from experience gained in fixed environments into a mobile context [Greenberg,91]. Recently, researchers have begun to address the particular problems posed by the communications delays inherent in mobile environments [Dix,95] but, as yet, no firm results have emerged.

The factoring of the application into separate interface components on the surface is mirrored underneath in the application itself. The application is built on top of a distributed object oriented architecture provided by a distributed systems platform (with semantics similar to those provided by ANSAware [APM,89]). The actual platform used and its relationship to ANSAware is discussed in detail in chapter 6. Briefly however, the architecture provides a uniform computational model consisting of *objects* which provide *services* through well defined *interfaces* (see section 6.2.3). Such an architecture greatly simplifies the challenges of writing applications for distributed environments by providing distribution transparencies (such as those highlighted in chapter 1) which enable services to be located and accessed regardless of physical location. In addition, the architecture provides a uniform set of system services and mechanisms for intercommunication which permit interoperability of applications within heterogeneous networks and end-systems (satisfying the requirement highlighted in chapter 4).

The application tools are implemented as a set of RM-ODP compatible objects which offer specific services via interfaces. All intercommunication is provided by the platform through an RPC or *invocation* mechanism. Each instance of the application accesses the modules of every other instance through invoking operations on the service interfaces. Thus, by converting the platform to run on a new architecture, the application can be ported with only minor changes. Furthermore, as all interactions are via service interfaces and the invocation mechanism, an application module need only support the specified interface to interwork with other application instances. So, for example, one module could be replaced by an equivalent without affecting the remaining components (for instance, replacing a particular GIS with a faster one) providing it is capable of supporting the same interface. Section 5.3 considers the application components in more detail.

Legacy applications also require RM-ODP compatible interfaces to be able to interwork with the application. These interfaces could notionally be integrated into the legacy applications themselves. However, since the majority of these applications are no longer under development, a gateway application is normally required to ‘wrap’ the application and offer its functionality through a service interface. Attaching well defined service interfaces to these applications has an additional benefit: once the interface has been defined, new applications can be developed by the organisation which are then able to interwork with the existing range of legacy applications using a simple common mechanism, despite the heterogeneous networks and end-systems in use.

An alternative, and much explored, strategy for implementing collaborative applications is through the use of a common windowing system, such as X, with a multiplexer to share graphical information between hosts [Abdel-Wahab,91]. However, research has shown little positive experience of using standard applications in a collaborative context [Lauwers,90]. Moreover, using such a protocol in a mobile environment raises a number of problems [Kantarjiev,93]. For instance, users are constrained to view shared information at exactly the same level of detail, resolution, size and position on their displays (violating the premise of a configurable toolkit approach). Moreover, such an approach would impair interactions between mobile field workers and office based colleagues that are using more powerful fixed workstations with larger higher resolution displays (particularly in the case of control centre staff who are required to deal with multiple field engineers simultaneously). In addition, windowing systems such as X generate a significant volume of high-rate latency sensitive protocol traffic which is poorly suited to mobile communications (although researchers are attempting to improve the performance of the X protocol over reduced capacity fixed links [Fulton,93]). Lastly, tying all hosts within the conference to a single windowing system implies firstly, that all hosts will be capable of running the system (which infers a range of hardware and software limitations) and, secondly, that the tools being used collaboratively will also run under the windowing system.

5.3 Application Components

The complete set of computational objects that make up each instance of the application prototype are illustrated in figure 5.1.

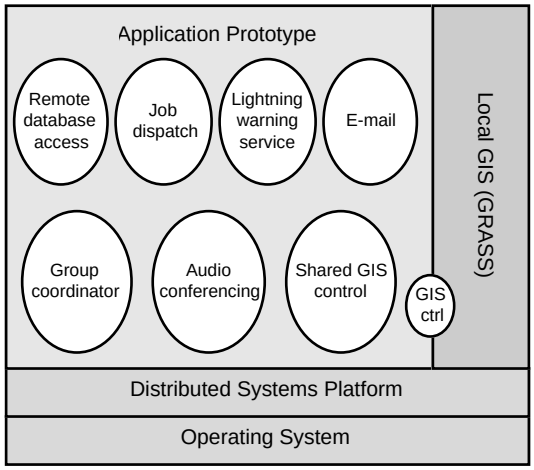

Figure 5.1 - Object structure of the application prototype

The following sections consider the design and implementation of each functional module in turn. The group coordinator module is considered first due to its central

role in supporting the group conferencing and state management needs of the other modules.

5.3.1 Group Coordinator Module

5.3.1.1 Overview

The group coordinator module is responsible for maintaining the decentralised group membership. The module also manages operations common to all group interactions, such as establishing the conference and dealing with dynamic changes in group state as members join and leave. In addition, the group coordinator module provides a stable, if pessimistic, fault recovery strategy where failed members are removed from the group by default, but may be reinstated by any user at any time. Mechanisms for bringing a user 'up-to-state' after a temporary disconnection from the group are provided. However, no automatic mechanism currently exists to bring a user's module specific views up-to-state (for example, synchronising the users GIS operations with that of the group), although this may be accomplished trivially with user intervention.

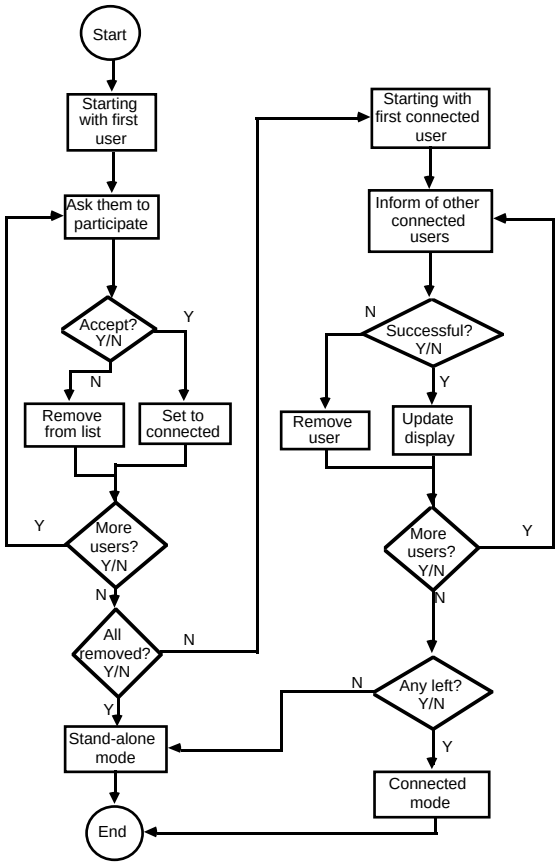

Figure 5.2 - Algorithm for starting a conference

The group control messages themselves are propagated via platform invocations. The algorithm for establishing a new conference is summarised in figure 5.2 (it should be noted that this is just a summary; the full algorithm contains additional complexity for dealing with establishing conferences with members who are already participating in conferences and so on).

The flowchart illustrates some of the complexity introduced by the possibility of failure during each communication phase. In essence, the algorithm consists of two passes. The first pass asks each prospective conference member whether or not they wish to join the conference. There are two possible outcomes: if they decline then they are removed from all further dealings with the group; if they accept then they may optionally inform the conference initiator of any further parties that should be included in the conference. The second pass is to inform all the users who assented to joining the conference.

The following sections examine the structure of the group coordinator in more detail. More specifically, the computational structure of the module, the RM-ODP compatible interfaces it provides to other modules and the module's graphical user interface are each examined in turn.

5.3.1.2 Computational Structure

The computational object structure of the group coordinator is represented in figure 5.3 [Davies,95b]. In the figure, interactions between objects are shown as numbered arrows.

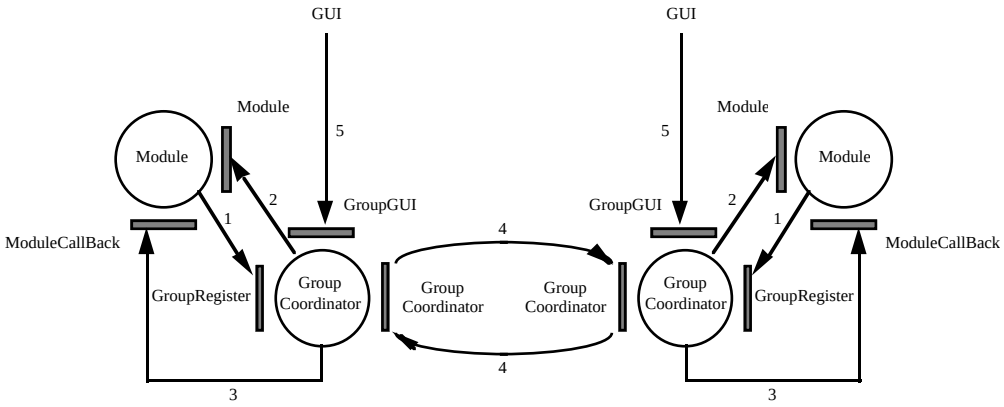

Figure 5.3 - Group coordinator object structure

The group coordinator maintains a pair of interfaces which offer private and public services to other application modules and to the applications running on remote hosts. The public interface is used by remote group coordinators to perform generic conference management operations such as connecting and disconnecting a user, adding users to an existing conference, and starting and stopping remote modules (via

interaction 1). The signatures of the public operations supported by the group coordinator are specified below (the examples throughout this thesis use the interface definition language specified by the platform, see section 6.2.5.1).

```
GroupCoordinator : INTERFACE =
BEGIN
ConnectRequest :    OPERATION [ user : User ]
                    RETURNS [ User; GC_Status ];
AddUser :           OPERATION [ user : User ]
                    RETURNS [ GC_Status ];
RemoveUser :        OPERATION [ user : User ]
                    RETURNS [ GC_Status ];
DisconnectRequest: OPERATION [ user : User ]
                    RETURNS [ GC_Status ];
StartRequest :      OPERATION [ userid : INTEGER; module : Module ]
                    RETURNS [ Module; GC_Status ];
AddModule :         OPERATION [ userid : INTEGER; module : Module ]
                    RETURNS [ GC_Status ];
RemoveModule :      OPERATION [ userid : INTEGER; module : Module ]
                    RETURNS [ GC_Status ];
StopRequest :       OPERATION [ userid : INTEGER; module : Module ]
                    RETURNS [ GC_Status ];
END.
```

Figure 5.4 - Group coordinator public interface

A private interface supported by each application module (interaction 2 in figure 5.3) allows the group coordinator to instruct the module to perform initialisation, terminate itself and show or hide its graphical interface. Modules are individually responsible for managing application specific communications between itself and its opposite number on the other hosts participating in the conference. So, for example, the communications module is responsible for establishing an audio conference between the participants. A private interface to the group coordinator (whose operations are specified in figure 5.5) provides a number of supporting functions. These functions include tests on the group state and a call-back interface for informing modules of changes in group membership.

```
GroupFunctions : INTERFACE =
BEGIN
TestGroupMode :    OPERATION [ ]
                   RETURNS [ BOOLEAN ];
InstigatorQuery :  OPERATION [ ]
                   RETURNS [ BOOLEAN ];
SetGroupMode :     OPERATION [ GroupOn : BOOLEAN ]
                   RETURNS [ BOOLEAN ];
RegisterJoin :     OPERATION [ Module : STRING; Callback : Interface ]
                   RETURNS [ Status ];
DeRegisterJoin :   OPERATION [ Module : STRING ]
                   RETURNS [ Status ];
RegisterLeave :    OPERATION [ Module : STRING; Callback : Interface ]
                   RETURNS [ Status ];
DeRegisterLeave :  OPERATION [ Module : STRING ]
                   RETURNS [ Status ];
ReportFailure :    OPERATION [ Module : STRING; Suspect : Interface ]
                   RETURNS [ Status ];
GetModuleId :      OPERATION [ Module : STRING ]
                   RETURNS [ Id ];
Fallback :         OPERATION [ UserId : CARDINAL ]
                   RETURNS [ ];
END.
```

Figure 5.5 - Group coordinator private interface

The application employs a straightforward mechanism for dealing with failure within the group. Individual modules can inform the group coordinator that a module they are in communication with cannot be contacted. Following a failure notification, the group coordinator will purge the specified module's interfaces based on the assumption that the remote component has failed (and consequently, its interfaces will have been invalidated, i.e. are now *stale*). At a later time the group coordinator will renegotiate with the remote group coordinator to obtain an up-to-date interface for the server once it has recovered. If catastrophic failure occurs, such as a remote node powering down or a detectable system error, then the fallback operation provides an expedient mechanism for removing that member from the group. More usually, group operations would not be propagated to that member until such time as they can re-establish communication.

5.3.1.3 User Interface

The group coordinator supports a graphical user interface which is shown in figure 5.6.

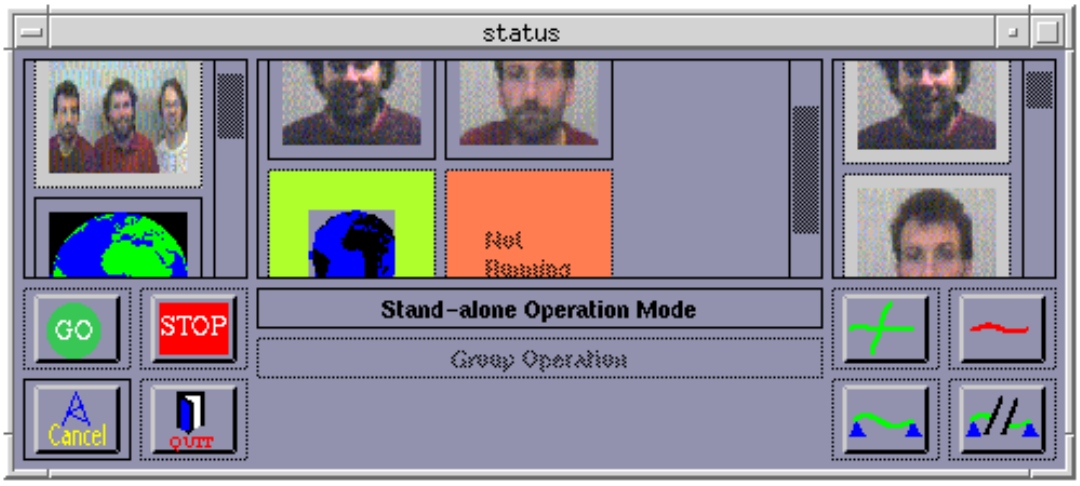

Figure 5.6 - Group coordinator graphical interface

The interface is pictured during a conference (in stand-alone mode the central display and right hand buttons are not displayed). On the left hand side is a scrollable list of icons which represent the modules that are currently available (the two shown are the conference manager and geographical information system (GIS), illustrated by a group photograph and a globe respectively). Underneath the list of modules are a set of module action buttons. These actions include: starting, stopping, quitting the entire application and, importantly, the cancel operation. The interface is underpinned by a state machine which guides the user through operations by highlighting and greying-out icons that are available and unavailable respectively according to the given state. For example, if the user is attempting to start a module running, they would click the

‘go’ button and the modules which are available and not already running would be highlighted. At any point the user may cancel and drop back to the ready state of the interface.

The right hand side of the interface has a scrollable list of potential conference participants. In the trial prototype the number of users was sufficiently low to make this form of scrolling list viable. However, in a real scenario it is expected that a more scalable solution would be required, such as a searchable index or browser. Below the scrollable list are the conference operation buttons: these allow the establishment and clearing down of conferences and the addition and removal of participants. Operations which affect other users require their consent before they may proceed. For example, clicking the remove button and then selecting a conference participant from the central panel will ask that member if they wish to leave, rather than removing them from the conference immediately. The one exception is ‘disconnect’: the disconnect button removes the user who presses the button from the conference; the act of pressing the button is taken as consent from that user to leave.

The central panel to the interface displays all the information relating to the state of the current conference. From left to right at the top of the window are the participants of the conference excluding the user whose display is shown (the central panel in figure 5.6 shows two other participants, implying that a three-way conference is in progress). In a list below each participant are icons representing the modules that are running on that participant’s machine (these are miniaturised versions of the module icons on the left hand side). The lists of modules are arranged such that corresponding modules on each host line up horizontally. In this case, the GIS module is running on the first participant’s machine but not on that of the second participant. These lists serve two purposes: firstly, the user can see at a glance the modules that the other conference members are using and hence which tools can be used for collaboration and, secondly, the module icons can be selected to toggle the propagation of module operations to that particular user. This permits the user to dynamically configure subsets of the group participants to receive operations from each of their modules. For example, an audio conference could be established between all the participants whereas GIS operations are exchanged by only two of the members. At any stage, the user may select globally whether operations on local modules are private or public to the conference (via the centrally positioned ‘group operation’ button).

The user icons at the top of the central window also allow the user to direct module operations to other users. For example, in the same way that the user would start their GIS by clicking first on the ‘go’ button and then on the GIS module, the

user might ask the other users for a GIS collaboration by clicking ‘go’ then one or more of the user icons and then the GIS module. The users who assented to the request would have their GIS modules automatically started and brought together in the conference providing the local GIS module is already running or can be started successfully.

5.3.2 Geographic Information System (GIS) Module

One of the central themes of the prototype is the sharing of spatially referenced data such as maps, schematic diagrams and engineering drawings provided by a GIS. Discussions during the requirements capture process identified a basic set of functions that the GIS module was to provide. In particular, the GIS module should support a shared workspace in which users can point, annotate or draw in a similar fashion to shared sketchpads or blackboard tools [Greenberg,91]. However, unlike these tools, the GIS module does not permit the transfer of images to limit bandwidth consumption. Instead, drawing operations are overlaid on a common base of locally cached images. All GIS module operations are propagated using platform invocations to satisfy the interoperability requirements outlined earlier.

The highlighting and map rendering operations are provided by a public domain GIS called the Geographic Resources Analysis Support System (GRASS [Westervelt,91]). The GIS has been modified to support two public interfaces: the first provides drawing operations (the IDL specification of a subset of these are shown in figure 5.7) and, the second, a number of operations for managing sets of figures.

```
DisFun : INTERFACE =
BEGIN
GetBox :     OPERATION [ ]
             RETURNS [ Coord; Coord; Coord; Coord; Status ];
DrawBox :    OPERATION [ BoxColour : Colour; x1 : Coord; y1 : Coord;
                         x2 : Coord; y2 : Coord ]
             RETURNS [ Status ];
DZoom :      OPERATION [ ]
             RETURNS [ Status ];
StartMon :   OPERATION [ Monitor : MonitorName; MonitorTitle : TitleText ]
             RETURNS [ Status ];
DRast        OPERATION [ RasterToShow : MapName ]
             RETURNS [ Status ];
DVect :      OPERATION [ VectorToShow : MapName; LineColour : Colour ]
             RETURNS [ Status ];
END.
```

Figure 5.7 - Subset of identified GIS operations

The public operations include operations for drawing raster and vectorised figures and for zooming in on particular areas identified by a standard ‘north-east’ geographical coordinate system (that can be easily converted to and from longitude-latitude coordinates). In addition, these operations have been chosen because they seem to have functionally equivalent operations in a variety of GIS systems (for

example, ArcInfo or SmallWorld), allowing different systems to be used at the implementers' discretion.

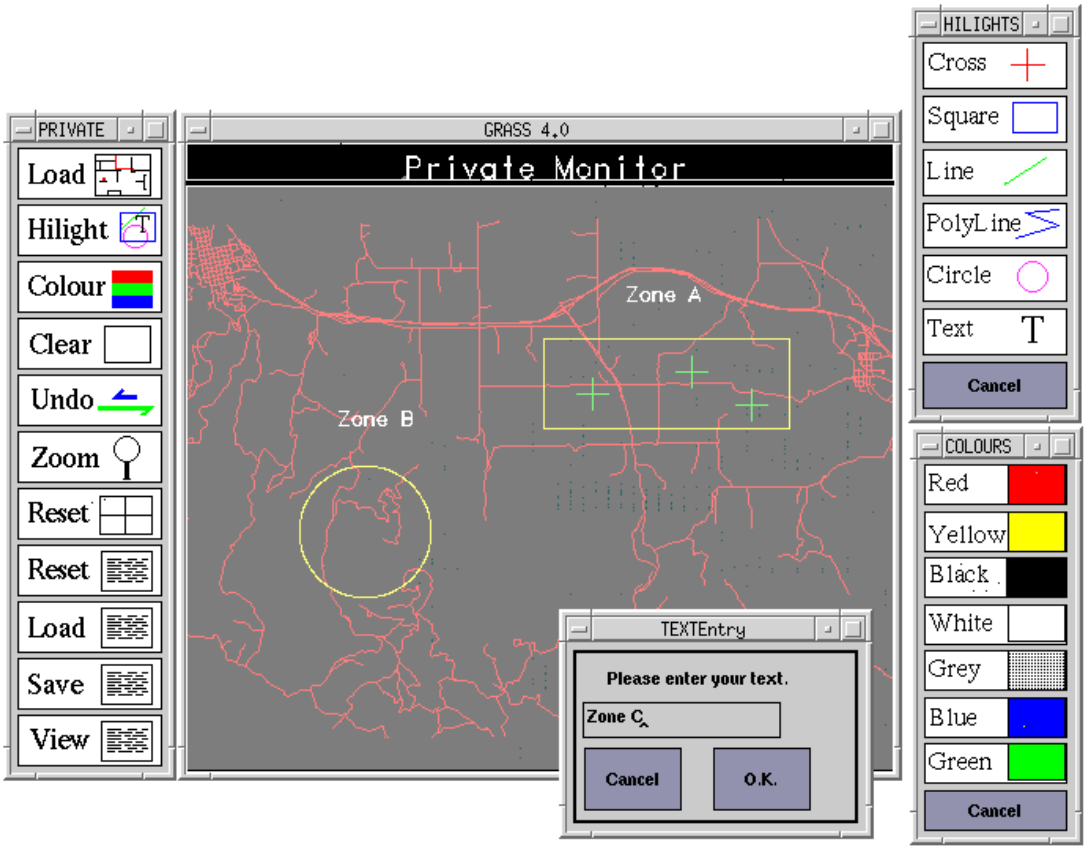

Figure 5.8 - The GIS user interface

The GIS operations are activated via the user interface to the GIS module shown in figure 5.8. The GIS supports two GRASS windows or *monitors:* a private monitor and a public monitor. The private monitor, as the name implies, can perform local GIS operations which are not propagated to other group members. The public monitor will perform the same operations collaboratively subject to 'group operation mode' being selected and members of the conference having their GIS modules functioning and selected. The two views give a clear distinction between information that is local to the machine and information that is globally accessible.

Figure 5.8 shows the private monitor together with the main set of operation buttons on the left hand side. On the right hand side are ancillary menus brought up from the main menu with buttons for selecting the current highlighting tool and the colour used for successive drawing operations. Each interface is an independent window object which may be scaled, moved or iconised to make the most of the limited display area. In addition to these tools, operations may be saved to a *clipboard* which can then be manipulated or replayed in either of the monitors. The clipboards allow the user to complete a set of operations 'off-line' in the private monitor and then

replay them en-masse in the public monitor to other group members. Additionally, clipboards may be attached to messages and sent asynchronously via the E-mail and job dispatch modules (see sections 5.3.4.2 and 5.3.4.3).

While the group coordinator module is responsible for generic conference maintenance functionality, the application specific knowledge of the GIS module is required to propagate public operations to the GIS modules of other users. Consider the scenario where a user wishes to focus the regions viewed by the other conference participants to highlight a particular section of schematic. The zoom operation on the local display causes a 'rubber-band' style box to appear on the relevant monitor to allow the user to drag and select the region to focus upon. Multicasting the operation to other group members would cause the same selection box to appear on the other conference participants displays which is not the intended function. Instead, the GIS module knows that, once the local zoom function is complete, the other group members' GIS modules should be told to select a particular map region via another interface operation. Such intervention requires application specific knowledge and hence collaboration aware group applications.

5.3.3 Audio Communications Module

5.3.3.1 Motivation

As highlighted in chapter 4, field engineers require audio communication facilities in order to be able to interact with the control centre and coordinate activities with their colleagues. In addition, audio communication support within the application prototype is essential for two further reasons :-

Conference management

In applications which provide data only interfaces between cooperating parties, the system is responsible for providing "floor-control" protocols for governing how a user's operations affect the state of the group. For instance, one user might be designated the "speaker", their operations will take precedence until they yield the floor by nominating another user. The tasks undertaken by field engineers are highly diverse, varying in response to the demands of a given scenario. The application is based on the premise that providing a data-only interface to engineers faced with a wide range of non-deterministic tasks would be too constraining. Instead, by providing audio communication facilities a social protocol between the engineers can determine the most appropriate form of dialogue and coordinate group interactions.

Safety criticality

> Field engineers, by their very nature, are required to work with voltage distribution networks which pose a threat to their safety. Currently, it is only through physical security mechanisms (such as those described in section 4.2.3) and verbal dialogues with the control centre, that potentially hazardous inconsistencies in network state are discovered. Audio dialogues contain a wealth of implicit safety and state information and are essential for providing context to collaborative data operations. Furthermore, experience within the utilities industries regards audio communication as inherently safer (and of course, more familiar) than data communication.

The remainder of this section describes the supporting components of the audio communications module in more detail.

5.3.3.2 Base Services

The audio conferencing module is responsible for establishing group audio communication between selected conference participants. The module relies on a set of multimedia transport services to provide the underlying management and transmission of the audio information. These services, known as the *Base Services* are based on earlier work at Lancaster University on developing a set of RM-ODP compatible services for managing the transmission of continuous multimedia data [Coulson,92] .

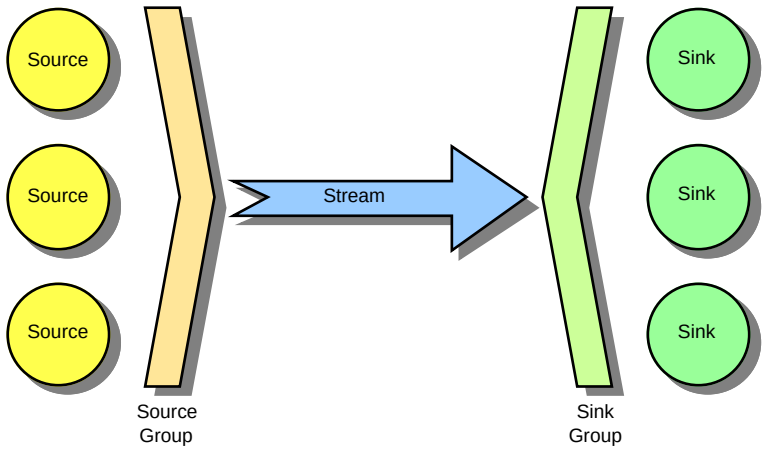

Figure 5.9 - Conceptual overview of the Base Services

Conceptually, the Base Services consist of three components: devices, groups and streams (illustrated in figure 5.9). Devices provide uniform abstractions for providers (sources) and consumers (sinks) of multimedia data, for example, sound capturing or video digitising hardware. Source and sink drivers are collected together to form source and sink groups respectively, each of which is controlled by a group manager

object. The group managers provide a level of abstraction over the underlying multicast and multidrop functionality. A computational object known as a stream object is responsible for organising the actual transport of information from sources to sinks and, in addition, is responsible for the selection and interposition of translators to provide filtering and mixing of the appropriate media type as necessary.

Continuous media has certain common characteristics. In particular, a given media type consists of discrete components or packets (a unit of audio or frame of video) which, when presented in a particular order within certain time constraints, represent a flow of information. This common or generic functionality has been singled out in the base services: each flow of information is regarded as a *chain* of media *links*. Operations common to all media types (and therefore all chains) include starting and stopping the flow of links and, in the case of finite length sequences, locating one's position within the chain and seeking to a new position. These operations are supported by RM-ODP compatible chain interfaces which are exported by device objects. The interconnection of device objects is accomplished using endpoint interfaces which provide common interconnection functions for underlying transport management by the stream object.

```
  /* Connect source and sink groups to stream */

! {s_status} <- streamIf$ConnectSource(sourceGrpIf)
! {s_status} <- streamIf$ConnectSink(sinkGrpIf)

  /* Set QoS Parameters */

! {QoS, s_status} <- streamIf$GetQoS()
  QoS.LinkSize = LinkSize;
! {oldQoS, s_status} <- streamIf$SetQoS(QoS)

  /* Prepare and commit stream object with QoS */

! {s_status} <- streamIf$Prepare()

  /* Prepare, commit QoS at source and start... */

  if (s_status == s_okay) {
!   {s_status} <- streamIf$Commit()

    if (s_status == s_okay) {
!     {c_status} <- sourceChainIf$Prepare()

      if (c_status == c_okay) {
!       {c_status} <- sourceChainIf$Commit()

        if (c_status == c_okay)
!         {c_status} <- sourceChainIf$Start(0, c_asynch)
     }
   }
  }
```

Figure 5.10 - Initiating a stream of media in the Base Services

The stream object provides an abstraction over the actual connection set-up and management. The stream management interface provides operations to allow groups of sources and sinks to be interconnected and supports a two-phase commit process which allows the communication to be aborted if the requested combination of drivers is unworkable (for either compatibility or connectivity reasons). A segment of code

illustrating basic connection establishment using a stream is shown in figure 5.10 (the platform's distributed programming language is described in more detail in section 6.2.5.1). The abstraction provided by the stream object would enable alternative media transport protocols to be selected transparently. For example, the current implementation does not make use of multicast (due to operating system compatibility issues within the chosen platform). Multicast IP could be slotted in to the stream object without affecting the remainder of the audio conferencing architecture. Conference membership can change dynamically through the group management interfaces. Membership changes are propagated to the stream object via a call-back interface which allows it to re-configure. For more details on the complete Base Services architecture see [Coulson,92].

A subset of the original Base Services has been reimplemented by the author. The reimplemented services use the new platform and provide audio transport services for the communications module. More specifically, these services include: the stream object, group manager, and audio chain source and sink drivers (both for transient and persistent chains). In addition, the services have been extended to include support for multiple audio stream formats (see section 5.4.4).

5.3.3.3 Audio Conference Management

Each host within an audio conference requires an identical set of base service objects. Viewed as a whole, the object configuration is identical and symmetrical across all participating hosts.

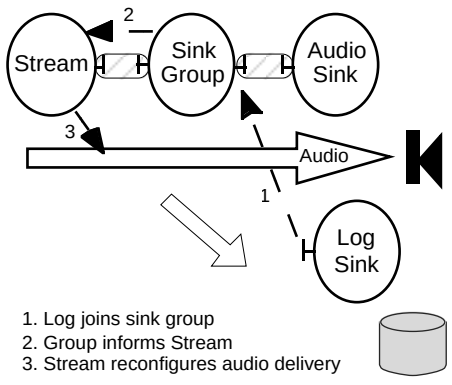

Figure 5.11 - Reconfiguring audio to add a persistent log

The objects are by default: a microphone driver, a speaker driver, a source group manager, a sink group manager and a stream manager. The driver objects simply provide two interfaces on to the audio sampling hardware of the host. The group objects allow dynamic reconfiguration of the conference (see figure 5.11), for example adding a logging device to record all outgoing or incoming audio traffic by

simply joining it to the appropriate endpoint group. The stream manager looks after the actual distribution of the audio packets.

The communications module makes use of the fact that the application, and hence the communications module itself, is replicated on each host within a conference. Since all membership changes are propagated to all reachable members by the group coordinator, the reconfiguration process is also distributed across all members, each taking an identical role. For example, adding a new member requires each communications coordinator to obtain the endpoint of the speaker driver of the new member and the inclusion of this endpoint within their local sink group. Since this process is executed by all conference participants in unison, the new member's speaker driver simultaneously joins the sink group of all of the initial participants. The removal of a member is an almost identical process where the member's endpoint is simply removed from the sink groups.

5.3.4 Additional Modules

The modules described in the previous sections have covered the most important collaborative tools within the application prototype. The remaining modules offer less significant tools such as remote access to legacy applications and asynchronous communication facilities (e.g. E-mail and job dispatching).

5.3.4.1 Remote Database Access Module

The remote database access module is a more traditional client-server tool which provides access to a simple database server which may be running on some remote host. The module was designed to demonstrate that applications running on mobile machines can interwork with existing database engines running on legacy systems within the established infrastructure, thereby proving the feasibility of access in the field to customer and resource databases (and further demonstrating the interoperability aspects of the work).

The remote database access module is a non-collaborative tool. While multiple users may be running the module simultaneously and each effecting queries to the remote database server, there are no facilities for sharing the query results between group members. However, it would not be difficult to allow the results of the query to be transferred to one of the other tools such as the E-mail module (see below) for dissemination to other engineers.

5.3.4.2 E-mail Module

The E-mail module permits the sharing of multimedia MIME messages [Borenstein,93] between users of the prototype application. The E-mail module is the only part of the application which has not been engineered as an RM-ODP compatible object in its own right (i.e. no interface is offered to applications running on the local or remote hosts). The lack of integration is purely due to time constraints within the MOST project: it was recognised as being more important to be able to demonstrate the mobile E-mail concept rather than spend additional time on an integral solution.

In essence, the system consists of an *elm* mail browser front-end which is supported by an RM-ODP compatible module in place of the SMTP daemon. The new module uses platform invocations to provide the mail transport mechanism rather than SMTP. This strategy avoids the need to use a mobile IP implementation or make any modifications to the mailer. The E-mail system could be extended to include a gateway object within the fixed infrastructure which transferred platform invocations containing E-mail messages to the standard mail daemon (and vice versa). Such an object would permit the global Internet E-mail system to interwork with the prototype's E-mail tool.

5.3.4.3 Job Dispatch System

The job dispatch module is designed to provide field engineers with a convenient mechanism for receiving work instructions and reporting job completion to the job issuer. The module is designed to provide functionality analogous to the paper based record system currently in use within the utilities companies. The job dispatch module, like the remote database module, is non-collaborative. Jobs are destined for a particular field engineer who may then, at their discretion, share information with their colleagues. However, the original engineer to whom the job was sent remains accountable for the completion of the job.

In addition to detailed instructions on the nature and location of a job, the system allows map and schematic references to be included with the instruction. The references can be imported into the GIS module so that the field engineer can view the job location in precise detail. An annotated instruction is inherently more precise than verbally delivered instructions from the control centre via the PMR system.

The job dispatch module relies on the same ODP transport service as the E-mail system for the reliable transmission of the job dispatch records.

5.3.4.4 Lightning Warning Service

The final module to consider in this chapter is the lightning warning service module. Lightning location is of vital importance to field workers in the electricity utility companies for a number of reasons. Firstly, since a large quantity of repair work is as a direct result of storm damage (in general) and lightning (in particular), knowing of the impending approach of a storm can allow the utility company to make strategic decisions about power routing and manpower in advance of a potential power outage. Secondly, where a field engineer is required to repair or service equipment, particularly if it is overhead or pylon mounted, knowing of the impending approach of lightning could potentially avoid life threatening situations.

The lightning warning service module is essentially a client front-end to a central service known as the National Lightning Flash Location System [Morgan,91] developed by the MOST project partners, EA Technology Limited. The service is a collation point for information from a number of listening posts established all over the U.K. By tuning in to the 800MHz frequency band, each listening post is able to detect the electromagnetic pulses generated by lightning strikes. The information is collated from a number of sites and triangulated to precisely locate each strike (the more sites involved, the more precise the location fix). The resulting lightning strike information (position and time of strike) is coordinated at a central site where it can be used to supply interested clients. Service subscribers receive strike reports at a rate of up to 5 strikes per second (40 strikes per second can be recorded centrally for later analysis). Subscribers normally run client software which overlays the strike information on a map of the British isles (see figure 5.12).

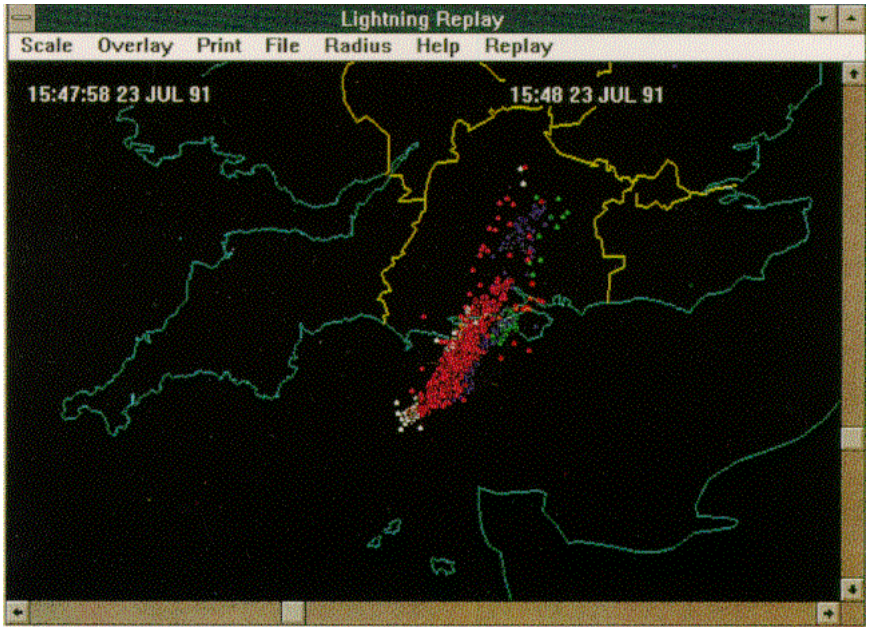

Figure 5.12 - Lightning flash location service

The lightning warning service module allows engineers to enter their location (a postcode is sufficiently accurate); the module registers with a remote server on the fixed network. The remote server parses the lightning strike information looking for incidences of lightning within a specified bound of its client engineers. Should lightning encroach within the threshold the engineer's module is called-back (via the registered interface) to inform the engineer of the approaching danger.

5.4 Implementation Issues

This section highlights a range of issues that emerged during the implementation and development of the application prototype. The hardware and software technologies chosen to underpin the application, associated constraints, and justification for these choices are outlined.

5.4.1 Hardware and Software Support

Phase one of the application and platform development work was conducted using an infrastructure of Sun workstations networked with 10Mbps Ethernet. These machines were chosen primarily because of the development team's familiarity with the UNIX operating system and the availability of an ANSAware distribution for SunOS 4.1. However, typical utilities use a wide variety of computer hardware, among which UNIX workstations are prevalent.

For phase two of the development, which was designated for migrating the office based software to the field, an Intel 486 PC workstation and a number of 486 notebook machines were used. The Novell UnixWare operating system was chosen for the PC development work because of familiarity with the UNIX System V Release 4 API. In addition, UnixWare was one of the first versions of UNIX available for the Intel chipset that supported an emulator capable of running Windows software. The emulator allowed the development team to retain the familiar UNIX API and meet the end-user's wish that Windows based applications (such as that used in their own mobile data project) should run.

The operating systems of the two platforms differed in an important respect: SunOS is Berkeley (BSD) UNIX, whereas UnixWare provides a System V Release 4 API. The two systems are largely compatible, though differ in innumerable small but significant ways. The project team were therefore required to maintain two versions of all software developed, compiled for both the architectures and supporting the API of each of the operating systems (the amount of common code was maximised through the use of conditional compilation).

Mobile communications support for the second phase was provided by cellular telephones. Initially, analogue voice channels were used with a cellular aware modem (conventional modems are prone to dropping connections due to the periodic delays caused by cell management procedures, particularly handover). Later in the project the GSM data service became available. The GSM service offered superior data rates and reduced connection times when compared with the analogue service, benefiting application performance. Early versions of the data service did not support mobile terminated data (i.e. mobiles could establish data calls, but not receive them), which was profoundly limiting.

The graphical interface for the application was written using the MIT X/Windows system Version 11 Release 6. The Athena widget set which accompanies X/Windows provided the basic facilities such as icons, buttons, scrollbars and so on. The public domain *pixmap* library was used to extend the widget set to allow colour pictures and icons to be created. The distributed systems platform used is considered in detail in chapter 6.

5.4.2 Group Working Issues

The prototype application is constructed independently of the underlying communications technology by using the generic platform invocation mechanism to perform all communication with remote objects. In the current implementation, group interactions are entirely specified in terms of these unicast platform invocations. In the fixed networking environment, few constraints are placed on the platform; each host is reachable without the need to consider the implications of the supporting communications medium.

In a mobile environment and, more specifically, if a connection-oriented technology such as GSM is used, the application programmer must consider the implications of high level actions on lower layers. In order to deliver invocations, the infrastructure will be required to establish and drop connections to remote hosts (either transparently or with the assistance of mobility-aware applications). Therefore, the number of conference participants and the order in which invocations occur can seriously affect the application performance.

To illustrate the point, consider the scenario in which an application is interacting with two applications on remote hosts A and B respectively (a three party conference). If the application is structured so that service invocations are interleaved (ABAB), lower layer connections are being dropped and established to deliver each invocation in turn (incurring the maximum possible latency for each invocation). If the application programmer is aware of the communications technology, the structure

can be changed to help improve performance. For instance, if there are no interdependencies between the invocations, then the invocations may be performed to each host consecutively (AABB), offering considerable performance benefits. In addition, if the system is capable of optimising the dialling strategy (by for example, batching invocations destined for the same host), then the application might use asynchronous invocations or perform both pairs of invocations concurrently to achieve the same performance gain.

Currently, no wide-area communications technologies support group data services, although emerging services such as CDPD, TETRA and GPRS may offer such facilities in the future. Group data service support is a requirement for effective group working with conferences consisting of more than two participants. In the MOST trial demonstrations this problem was circumvented by using a machine with multiple dial-up connections as an exchange. The exchange accepted multiple connections simultaneously and routed platform RPCs over the appropriate point-to-point connection according to the specified destination. This solution is limiting in terms of scalability and has a detrimental impact on the performance of the system.

5.4.3 Audio Conferencing Issues

As previously discussed in section 5.3.3, the audio conferencing module relies on audio transport services provided by the Base Services. The Base Services use the UDP protocol to transmit the packets of audio over the fixed networked environment. In fixed environments, such as an Ethernet, it is acceptable to transmit audio packets without compression. A single stream of audio traffic comprised of 8 bit samples, sampled at a rate of 8KHz will take 64Kbps of network bandwidth. Clearly, in a wide-area mobile context (which currently is likely to offer an absolute maximum throughput of 19.2Kbps) this volume of information is impractical.

One possibility for reducing the bandwidth requirements of a multimedia stream is through application specific encoding techniques. For instance, the peculiarities of spoken language allow the voice traffic to be *low-rate encoded* which reduces the bandwidth consumption by providing varying degrees of fidelity. The data rate can be reduced substantially over normal audio traffic due to the human brain’s ability to reconstruct damaged information into intelligible speech. Generally such techniques are highly CPU intensive, requiring dedicated coding hardware. In particular, digital telephone systems such as GSM employ voice encoders which operate nominally at 13Kbps. Research quality systems exist which can transmit recognisable, if slightly impaired, speech at 1.2Kbps. It should be noted that the rate at which a voice can be encoded depends very much on the source stream. For example, male voices generally

stand a higher rate of coding than female voices, and tonal languages such as Korean are particularly difficult to encode successfully.

For the purposes of the MOST project, the issue of how to transmit audio over low bandwidth channels was considered the responsibility of other fields of research. In actuality, cost and size implications of using a prototype low-rate encoder dictated that an alternative solution was found. Therefore, during the end-user field trials, two telephone handsets were used for each portable: one for data and one for audio traffic. Once the encoder hardware becomes viable, there are clearly a number of issues which require resolution. In particular, how voice and data traffic should be multiplexed onto a single channel whilst preserving the continuity of the audio, giving sufficient responsiveness to interactive traffic and avoiding starvation of lower priority traffic. These issues are not addressed in this thesis.

5.4.4 Audio Mixing and Format Conversion

The requirement that the application should promote interoperability in a heterogeneous environment has obvious implications for the platform and the transmission of platform invocations. However, heterogeneity of hardware has an additional impact on the transmission of multimedia information and, in particular, audio. More specifically, each hardware platform chosen for the project uses different audio sampling and encoding technology. The Sun workstations have on-board sampling hardware which produces a stream of samples in μlaw encoded format. The μlaw encoding technique was designed to encode audio for transmission over analogue telephony systems. Each 8 bit encoded sample is equivalent to a 13 bit linear sample through a non-linear translation (some detail is lost at lower volume where human hearing is not as sensitive). The PC workstation has additional hardware which can support a number of encoding formats including 16-bit pulse code modulation (PCM) and μlaw encoding in mono and stereo. The portable PCs have different on-board encoding hardware again which provide a subset of the workstation PCs functionality. These on-board encoders can produce 8-bit PCM samples (chosen as the standard format for PCs as there are likely to be more portable PCs than workstations in any given configuration). A single mono channel sampled at a rate of 8KHz was chosen as the default for all configurations.

In a conference situation, i.e. where more than two parties are involved, it is often necessary to mix packets of audio sent from the microphone drivers of more than one host. In the current implementation, this mixing may take place at any of the stream managers or speaker drivers within the interaction. Mixing is automatically triggered at an object by the arrival of packets from multiple sources. Each audio packet is

tagged with header information which identifies, among other things, the format of the audio data. In a one-to-one interaction, the audio packets will be sent directly from the source to the sink with no additional mixing stages (a format conversion will be applied only if the ends have different native formats).

To allow the base service drivers to interwork in such an environment it was necessary to provide a number of format conversion routines. The driver objects and stream manager are capable of providing a number of format conversion services. In the cases of the speaker driver and stream manager objects in particular, the conversion services are integrated with the mixing functionality to allow multiple formats of source information to be mixed into a desired destination format.

The stream manager attempts to determine the optimal number of mixing stages within a given set of source and destination endpoints to reduce the overall complexity of the conversion processes. The stream manager can make three optimisations :-

- If there is a single source and multiple destinations where the source has a different format to the sinks, then the source is told to send in the destination format.
- If all destinations and the majority of sources are of PCM format then the stream can tell all μlaw sources to send in PCM format (which allows an optimised mixing algorithm to be used downstream).
- The stream object works out which format among its sinks is most prevalent and then ensures the mixed output is in the more common format.

5.5 Summary

In this chapter the prototype application which addressed the requirements set out in chapter 4 has been discussed. The prototype consists of a number of application modules providing, among other things, a collaborative GIS for sharing geographic and schematic information, an E-mail system, remote database access and audio conferencing facilities. The application is built upon a mobility-aware distributed systems platform developed by the author and which is presented in detail in chapter 6. The application makes use of particular aspects of the platform to gain information about the underlying environment. With this information the application is able to adapt its behaviour in response to changes in its environment. The implications of mobility and application adaptation are considered further in chapter 7.

Chapter 6

Mobile Infrastructure

6.1 Introduction

This chapter explores the issues of providing infrastructure support for advanced mobile applications, particularly within heterogeneous networked environments. A specific distributed systems platform† which provides this support is discussed in detail. The mobility support within the platform is based on a number of computational model extensions, aligned with the RM-ODP framework [ISO,95a]. The RM-ODP is used as the basis for discussing the mobility enhancements to the platform.

Section 6.2 describes the basic architecture upon which the new platform is based. A number of shortcomings are identified within this architecture which prevent its operation in a heterogeneous mobile environment. Section 6.3 highlights the role of QoS in the support of advanced mobile applications and presents a set of extensions designed to address these shortcomings. In particular, support for explicit bindings have been added to the platform to enable QoS management, which in turn, allows application level adaptation. The computational binding architecture is discussed in section 6.4. Engineering level support for the QoS extensions, including a new RPC protocol called QEX, is detailed in section 6.5.

† Sometimes called a *middleware* platform, in recognition of its mediating role between the application and system software.

6.2 Choice of ANSAware

6.2.1 Rationale

To meet the requirement for interoperability within heterogeneous networked environments, the application prototype requires support from a distributed systems platform. The platform is based on an earlier distributed systems platform, known as ANSAware [APM,92]. The ANSAware platform is a partial implementation of the Advanced Networked Systems Architecture (ANSA) [APM,89], which in turn, was profoundly influential in the specification of RM-ODP.

An RM-ODP based platform was chosen over the competing OSF/DCE and CORBA models, for the following reasons :-

i) Both the ANSAware platform and the RM-ODP are well established and offer elegant object-oriented paradigms which facilitate well structured, modular design. In addition, the frameworks provide traditional object-oriented programming benefits such as encapsulation, type hierarchies and code reusability.

ii) DCE provides a less flexible client/server based architecture which enforces particular technological choices on developers. For instance, client/server interaction is through a specific RPC mechanism (based on Digital's Network Computing System).

iii) The source code to implementations of the CORBA platform were not available for experimentation.

It is important to note that while a particular platform was chosen as the starting point for development, the concepts highlighted in this thesis are intended to be generally applicable to other platforms.

ANSA and its engineering realisation, ANSAware, are described in more detail in the following sections. A number of the differences between the RM-ODP and ANSA frameworks are emphasised.

6.2.2 The ANSA Project

ANSA is the result of research conducted by a large industrial consortium under the auspices of the U.K. Alvey research initiative. The architecture comprises five projections, which are termed enterprise, information, computation, engineering and technology. Different facets of a given distributed system are revealed by viewing the system using any of these projections (a particular view of a system is termed a

model). For example, the resulting technology model would explain how the system maps on to computers, networks and software systems.

The computational and engineering projections are of particular relevance to this discussion. The computational projection describes how a system's software architecture is designed and structured. The engineering projection describes how the theoretical architecture specified by the computational model is realised in implementation terms. These two models are each considered in turn† .

6.2.3 Computational Model

The ANSA Computational Model describes a system as being comprised of one or more component *objects*. All data is encapsulated within objects and accessed indirectly using the object's *interface*. An object may provide or make use of multiple interfaces. Each interface is defined in terms of a set of named operations and a property specification. Each operation has an associated list of typed arguments which specify the parameters passed with each request and reply and a set of attached properties. The properties specify what transparencies and constraints are associated with an action or with the whole interface.

Operations are *invoked* via references to their enclosing interface (to maintain access transparency). Two forms of invocation are defined :-

Synchronous

> The client blocks until the server has performed the requested operation and delivered the results. The two objects are synchronised by the interaction.

Asynchronous

> The client does not block. Furthermore, there is no confirmation that the operation has terminated (or even begun). Asynchronous operations minimise latency and maximise concurrency where immediate results are not required (or no results are required at all).

An operation may be able to return one of a number of named responses or *terminations* (each potentially with multiple arguments). A termination may be raised to indicate failure and change the sequence of actions taken after the invocation. The one exception is the unnamed termination which can be thought of as the normal response from the operation. Partial failure could result in inconsistent state and orphans (operations which execute after the invocation is terminated). Therefore,

---

† The details presented in the following sections are as found in [APM,89].

ANSA defines invocations as being atomic (to the client, invocations either succeed or fail; the server must manage any inconsistencies due to partial failure itself). The success of asynchronous operations can only be determined through serialisation with synchronous operations.

In order that a client can stress the urgency with which it requires a server to have completed an operation, a deadline may be set. There are two forms of deadline: soft deadlines and hard deadlines. Soft deadlines only affect the scheduling of an operation. Hard deadlines will also prematurely terminate the operation once the deadline has been reached.

Objects may freely exchange *interface references*. Possession of an interface reference is sufficient to enable that object to access the operations described by the referred interface. Objects may publish interface references to their services with a system directory or name-service known as the *trader* (references so published are referred to as *offers*). The trader itself may be a computational object whose location is well known, or alternatively, may be a decentralised architectural service which is maintained by broadcast algorithms. Client objects can obtain the use of services by querying the database maintained by the trader and obtaining an offer (see figure 6.1).

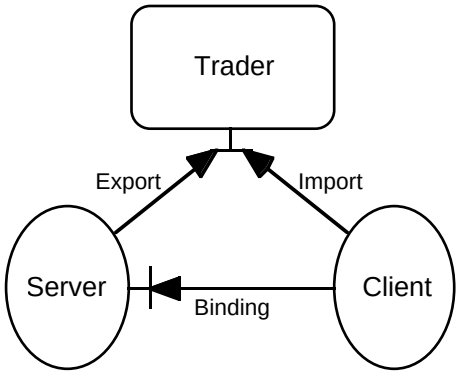

Figure 6.1 - Service location using the trader

An *implicit binding* is established when the service is first invoked. Thereafter the client can be referred to as being *bound* to the interface.

6.2.4 Engineering Model

The engineering model (see figure 6.2) describes the realisation of a system over multiple technology models. Conceptually, objects at the computational level are mapped onto engineering level objects to provide application and architectural services. Computational objects together with a *nucleus* object, *transparency*

mechanism objects and an *interpreter* (which enables interactions between computational objects) are known collectively as a *capsule*.

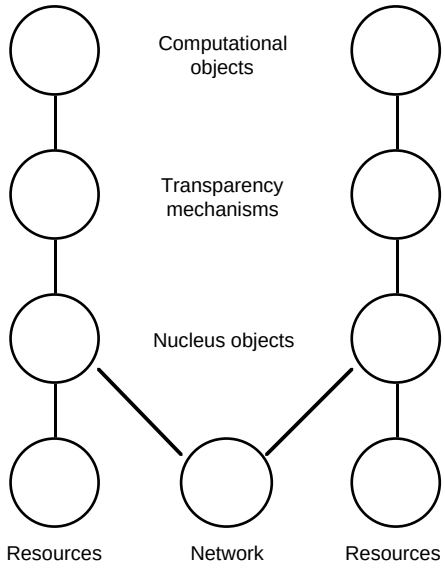

Figure 6.2 - ANSA Engineering Model

The nucleus is an engineering object which encapsulates the heterogeneity of processor and memory architectures. In addition, the nucleus provides a set of functions for providing concurrency control and enabling access transparent interactions between computational objects. The nucleus comprises a number of factory objects which embody aspects of functionality, for example, thread scheduling or capsule management (e.g. the instantiation of a new capsule from an object template). These factory objects are designed to be single threaded and interact using standard procedure call semantics.

All parallelism within the nucleus is handled by the *scheduler*. Parallelism is described in terms of threads, tasks and evaluators. An evaluator object represents a CPU within a given system. Evaluators each run a thread associated with an ongoing task (tasks represent the context of an ongoing computation and are created at the first context switch). The thread factory includes provision for the pushing and popping of deadlines for controlling the scheduling of the current thread. Soft deadlines are purely advisory and no action is taken on expiry. However, expiry of hard deadlines should cause the scheduler to notify the task associated with the thread and cancel ongoing remote interactions. In effect, a termination corresponding to a deadline expiry will be generated.

Concurrency constraints on interface invocation are maintained by a prologue and epilogue that bracket the execution of an operation. Mutual exclusion is ensured by *eventcounts* and *sequencers* [Reed,79]. In addition, these synchronisation primitives are made available for general application use.

Remote interface invocations are transmitted using an RPC based model. The RPC paradigm was chosen based on the assumption that bursts of simple interactions will be more common than sustained bulk data transfer. The protocol is designed to consume a minimum of local resources (buffers and timers etc.) and multiplex interactions over end-to-end network connections to maximise the potential concurrency of the system. To maximise protocol independence, the network aspects of the engineering model are split into three layers :-

Message passing services

The message passing services (MPS) manage connection, disconnection, sending and receipt of messages. All message passing services offer the same interface.

Execution protocols

The execution protocols map computational invocations onto message exchanges which are transferred using the message passing services. All execution protocols conform to the same interface.

Session layer

The session layer coordinates execution protocol and thread interactions. Any local state concerning remote interactions is stored within this layer.

Remote interactions require a plug and a socket to be established at the local and remote ends respectively. The combination of execution protocol and message passing service can be specified for each plug/socket pair. The local plug interface offers the operations shown in table 6.1.

Operation	Function
call	perform blocking request-response interaction (synchronous invocation)
cast	initiate announcement (asynchronous invocation, no response)
request	initiate non-blocking invocation (guaranteed delivery of service invocation)
collect	complete non-blocking invocation (collect or block for results of previous operation)
rebind	enable the service to be mapped to a new host (for service migration)

Table 6.1 - Operations for initiating invocations

The operations shown above map onto the synchronous and asynchronous invocation primitives specified by the computational model. The plug/socket

interaction together with the session layer state can be thought of as an implicit computational binding.

6.2.5 ANSAware

The ANSAware distributed systems platform is an instantiation of the ANSA model with specific technology choices for each component. This section describes the ANSAware platform from computational and engineering aspects. Lastly, the section gives a brief overview of the architectural services supplied with the platform for object management.

6.2.5.1 Computational Aspects

The ANSAware platform supports the object-oriented architecture described above; services are encapsulated within objects which are accessed via interfaces. Object interfaces and service invocation require two new languages: the Interface Definition Language (IDL) and Distributed Programming Language (DPL).

The IDL language provides a platform independent method of describing the operational interfaces to application level services. The resulting interfaces maintain portability across heterogeneous platforms by describing each operation in terms of well defined ANSA types which map on to equivalent representations on each platform. A generic IDL example is shown in figure 6.3 (further examples of actual IDL code can be found throughout chapter 5).

```
IFTypeName1 : INTERFACE =
[IMPLEMENTATION] IS COMPATIBLE WITH IFTypeName2 [FROM File];
NEEDS IFTypeName3 [FROM File];
BEGIN
  -- Interface-specific data types

  Status : TYPE = {if_okay, if_notOkay};

  -- Operations

  OpName : OPERATION [ arg1 : type1; ... ; argM : typeM ]
             RETURNS [ rtype1; ... ; rtypeN ];

  AnName : OPERATION ANNOUNCEMENT [ arg1 : type1; ...; argM : typeM]
                          RETURNS [ ];
END.
```

Figure 6.3 - A generic IDL interface

As the example specification above illustrates, an IDL can be defined as being “COMPATIBLE WITH” another interface, allowing type hierarchies to be constructed. In addition, defining an interface as having a compatible implementation enables the actual service code to be inherited (enabling code-reuse). Specifying an interaction action as an OPERATION, or more correctly as an interrogation operation, implies that synchronous service invocations are permitted on that operation. The

keyword “ANNOUNCEMENT” specifies that the operation accepts only asynchronous casts (see section 6.2.3).

The ANSAware interfaces do not provide support for action or interface properties as defined by the computational model. Consequently, ANSAware supports both access and location transparency, but provides no mechanism for selectively adding or removing transparency functions.

The IDL language is compiled into a set of engineering level transparency services known as *stubs*. The stubs convert the ANSA typed arguments into hardware dependent representations and provide the marshalling and unmarshalling facilities for RPC arguments so that they may be transported by the MPS functions.

The DPL language provides a common mechanism for embedding the invocation of object services within a given host language (see figure 6.4). Currently, bindings for the C, C++, Modula-3 and Ada host languages are available. The embedded DPL statements are translated into the host language using a pre-processor.

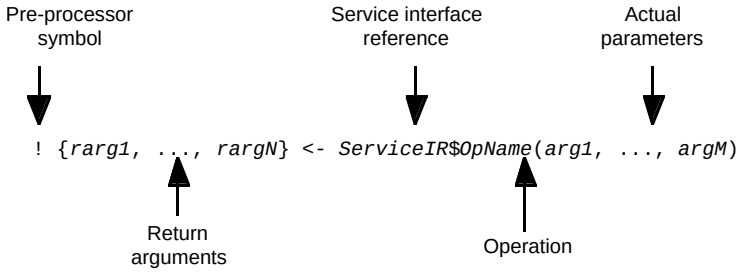

Figure 6.4 - A generic embedded DPL service invocation

Simple pre-processor directives allow the compiler to attribute interface types to program variables and allow the program to dynamically create and destroy interface references. The pre-processor is subsequently able to provide rudimentary type checking such as validating the number and type of arguments passed in invocations. The pre-processor does not have any knowledge of the host language and as a consequence does not obey the variable or type scope rules that are normally imposed. The remainder of the pre-processor directives are shown in table 6.2. A full explanation of the IDL and DPL languages can be found in [APM,93].

Directive	Function
`! USE TypeName`	Source depends on IDL *TypeName*
`! DECLARE {vars} : TypeName CLIENT | SERVER`	Attribute *vars* as interface reference to service *TypeName* (CLIENT or SERVER, but not both)
`! {var} :: TypeName$Create(concurrency)`	Create interface reference which will service *concurrency* number of concurrent invocations
`! {} :: TypeName$Destroy(var)`	Destroy the interface reference associated with *var*
`! {v} := ServiceIR$OpName(args)`	Initiate a non-blocking invocation (returns voucher *v*)
`! {results} <- ServiceIR$Redeem(v)`	Complete non-blocking invocation (associated with voucher *v*)
`! {ref} <- traderRef$Import(type, context, constraints)`	Query name-service for offer
`! {ref} <- traderRef$Export(type, context, properties)`	Export service offer to name-service (with associated *properties*)
`! traderRef$Withdraw(ref)`	Remove offer from trader

Table 6.2 - Summary of DPL pre-processor commands

ANSAware supports both blocking and non-blocking service invocations (implementing synchronous and asynchronous invocations respectively). Non-blocking invocations are often referred to as *vouchered* invocations (a voucher is returned from the initiation which is required for result redemption). Support for soft and hard deadlines, as defined by the computational and engineering models, are not supported in ANSAware (some support for these forms of deadline have been incorporated into the new platform, see section 6.4.3).

6.2.5.2 Engineering Aspects

The application (DPL code) together with the IDL generated stubs are linked with a supporting library which provides the remainder of the distributed systems functionality. The combined package is analogous to the engineering level capsule (in the UNIX domain a capsule corresponds exactly to a single executable process). The library and marshalling functions together provide the functionality of the nucleus.

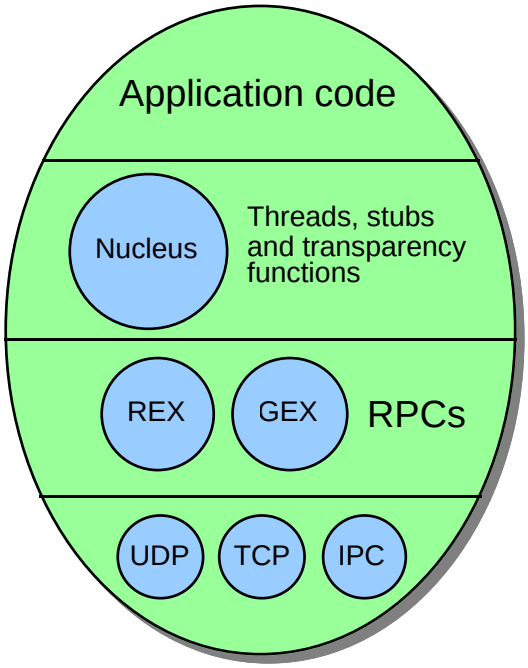

Figure 6.5 - An ANSAware capsule

An ANSAware capsule is illustrated in figure 6.5. The figure illustrates the layering of the primary nucleus functions, the execution protocols and the message passing services. The nucleus layer provides an architecture independent set of facilities (as previously outlined in section 6.2.4) including threads, stubs for marshalling and unmarshalling datatypes for the RPC mechanisms and functions which implement the distribution transparencies. The execution protocol layer includes two protocols: the remote execution protocol (REX) and, layered upon REX, the group execution protocol (GEX). These RPC mechanisms provide all inter-capsule invocation transport services (invocations to objects which are co-resident within a capsule are transparently converted to local procedure calls). The RPC protocols make use of transport services provided by the MPS layer. The MPS layer consists of a number of operating system independent interfaces to well known transport protocols such as UDP, TCP and IPC (named pipes). The interfaces between the nucleus, RPC and MPS layers are designed to be as generic as possible to enable support for new RPC protocols or MPS interfaces to be added without affecting higher layers (for example, a message passing service based on shared memory).

6.2.5.3 Architectural Services

The ANSAware suite of programs is supplied with four architectural services :-

Trader The trader provides the system directory service described in section 6.2.2. The trader is a centralised service which offers a service interface at a well-known address. Offers exported to the trader may

be placed within a hierarchical named context and may have a set of associated properties (name, value pairs). Traders may be federated to form a simple one layer hierarchy comprising a single master trader and a number of slaves.

Factory The factory object supports mechanisms which enable the dynamic instantiation and termination of service objects on a particular node. The factory requires a stored template (capsule executable) of the object it is required to instantiate.

Node Manager The node manager offers a range of services for automating the factory creation, monitoring and destruction of services on a single node. An alias for a service and an associated attribute can be logged with the node manager. The attribute specifies how that service is to be managed. For example, it may specify that the service is to be restarted on termination, or, dynamically instantiated when a client attempts to import the service's alias from the trader.

Storage The storage facilities enable services to be made persistent: that is, seamlessly moved to and from backing store (passivated and activated respectively) on demand. The storage system consists of two components: a snapshot database and a storage database. The former component stores object state while the object remains passivated. The latter component records a database of interface reference to service mappings, enabling a request for a passivated object to be recognised and mapped to the new interface reference on activation. Furthermore, service migration between nodes is supported (essentially a passivation on the source node followed by an activation on the destination node).

6.2.6 ANSA and RM-ODP Reference Models

The ANSA model lent many concepts to the formation of the RM-ODP. However, the RM-ODP model differs from ANSA in a number of important respects; including the number and types of object interface supported, how interfaces are specified, the causality of interfaces, support for QoS and forms of object binding. These variations are each examined in more detail below.

Firstly, the ANSA model supports only one type of interface which specifies the services that can be provided by an object. In contrast, the RM-ODP defines three

forms of interface: operation, stream and signal. Each operation type is outlined below :-

Operations An operation may be either an invocation or an invocation followed by a termination (corresponding to an ANSA announcement and interrogation respectively).

Streams A stream defines named flows which abstract over sets of interactions. These interactions are hidden below the interface level and are designed primarily for handling continuous media (e.g. audio and video).

Signals A signal corresponds to the occurrence of an event and is primarily designed to provide low level support to facilitate real time synchronisation and QoS management. For example, a signal can be generated on emission or reception of an invocation or flow.

The operational interface has the most similarity to the interfaces specified in ANSA.

Secondly, the ANSA computational model specifies that properties may be associated with operations and interfaces to govern the transparencies and invocation constraints enforced by the architecture. This concept is extended in RM-ODP through environmental contracts. Each interface type is defined by an interface signature which, in addition to named actions of the above kinds, specifies the causality of the actions and, potentially, an environmental contract (which includes QoS annotations). Interface signatures together with a behavioural specification and an environmental contract form an object template (which defines how an object may be instantiated by a factory). The environmental contract (either on a per interface or per object template granularity) enables the designer to impose constraints on the service usage. For example, the contract may specify forms of transparency that are to be enabled or disabled, or place recommendations for the final distribution of an application's objects.

ANSA interfaces always denote services that can be provided (operations accepted) by an object, and thus, all the operations defined by an interface have consumer causality†. As stated above, the RM-ODP interface signature allows the specification of the causality of each operation individually. This causality specification determines whether an action is produced or consumed (but not both). Note that, in order to implicitly (or explicitly) bind a pair of interfaces, it is necessary

† However, it is worth noting that the causality of an interface reference is determined by context.

that they have complementary causalities (an implicit binding to a server operation interface may cause a complementary interface to be generated automatically).

The ANSA model does not provide any mechanisms to enable the specification of the QoS produced or required by operations on an interface. In contrast, as mentioned above, RM-ODP environmental contracts can include QoS annotations. The QoS annotations consist of two clauses: the QoS provided to the environment and the QoS required from other services and the environment. Contractually, an object will be able to meet its provided level of service if and only if it receives the required QoS from its supporting objects (both computational and architectural).

Finally, the RM-ODP model offers a more comprehensive binding architecture than ANSA. In ANSA, an implicit binding is established to a service on first access, after obtaining an offer to the service from a trader directory service. RM-ODP also offers implicit bindings with the same semantics. Note that the trader service defined by RM-ODP offers a more flexible federation hierarchy, whereby traders may be contractually linked on a peer-to-peer basis. As well as implicit bindings, the RM-ODP also offers *explicit bindings*† for operation, stream and signal interfaces. In addition, there are two forms of explicit binding action defined: primitive and compound.

Primitive binding actions

A primitive binding action binds two computational objects directly, subject to the preconditions that both objects are of the same kind (i.e. operation, stream or signal) and have complementary causalities.

Compound binding actions

Compound binding actions may be formed by linking two or more objects via a binding object with primitive binding actions. The binding object is essentially an ordinary computational object with an additional set of formal role parameters in its object template (defined in terms of the interfaces to be bound). The binding object may apply object specific admission control techniques, such as QoS negotiation, during binding establishment. Furthermore, during bind time a set of control interfaces are generated which enable the monitoring of the binding, changes in the bound group membership and changes in the specified QoS.

† Real-Time ANSAware [Li,94] offers a mechanism called an explicit binding. However, this mechanism is designed to enable the selection of a real-time thread scheduling policy at the client endpoint and should not be confused with the explicit bindings mentioned throughout this chapter.

An explicit binding action may be invoked by any object, including third party objects. In addition, a binding action may incorporate point-to-point or multipoint-to-multipoint object bindings. Implicit multipoint-to-multipoint binding actions are possible in the ANSA model using the group transparency afforded by the GEX protocol.

6.3 Impact of Mobility on ANSAware

Chapter 2 presented the notion of integrated heterogeneous networked environments in which a user roams seamlessly between overlapping network technologies. Quintessential to such an environment are sudden and dramatic changes in network QoS. This section considers the impact of a dynamic supporting infrastructure on the ANSAware platform. In addition, a set of extensions are introduced which ameliorate the platform; enabling it both to operate in a changing environment and provide mechanisms which enable applications to adapt.

6.3.1 The Issue of QoS

ANSAware is a traditional distributed systems platform which exploits distribution transparencies to provide benefits to distributed application developers. However, as previously argued in chapter 3, the provision of distribution transparencies is not necessarily feasible or desirable in a mobile context. Moreover, the problems inherent in transparent mobility management are further exacerbated when operating in an environment which offers variable connectivity.

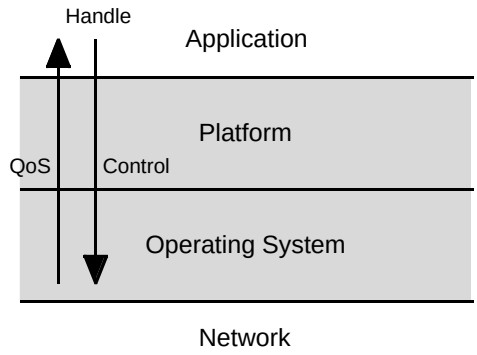

Figure 6.6 - QoS Architecture

In such cases, what is required is a framework which permits the flow of information from the supporting architecture up to the application and, in addition, offers a handle to applications in order that control can be exerted on lower layers (illustrated in figure 6.6). For instance, an application can communicate its requirements to the infrastructure and in turn receive information on the actual service

it is experiencing. Furthermore, if the expectations of the application can no longer be met (perhaps due to migration to a new wireless service), the application can be notified, enabling application specific knowledge to be applied (such as adjusting the requirements of the application, or the selection of an alternative communications service).

The necessity for a framework such as this, is a requirement for a QoS based architecture. There has been a considerable body of work in developing QoS architectures for controlling the flow and specifying the infrastructure requirements of multimedia streams. The next section considers the applicability of these established architectures within the context of mobile distributed systems platforms.

6.3.2 Aspects of QoS for Mobility

6.3.2.1 Rationale

Most of the existing work on QoS has concentrated on producing parameters which describe the characteristics of fixed networks. More specifically, QoS based protocols have been developed for transporting flows of continuous media, particularly within the high speed networking community [Delgrossi,93], [Campbell,93], [García,96]. A number of core QoS parameters have emerged as being useful descriptors for continuous media: typically these include throughput, latency, jitter and bit error rate. These features of the communications channel are of particular significance for dealing with the regular and timely delivery of the large quantities of information required by continuous media applications. For example, in the case of continuous video, the throughput must be sufficiently high to transport the frames of video, the jitter (variation in latency) low enough to ensure the frames are delivered within time bounds and the bit error rate below the threshold at which any video encoding technique employed on the media (such as MPEG compression) breaks down. With these QoS parameters, it is possible to reserve sufficient network and end-system resources to ensure the correct and timely delivery of the flow of information. Should the availability of the resources change, the parameters are sufficient to enable a QoS violation to be generated and the required QoS to be renegotiated.

In addition to continuous media transport requirements, distributed applications are often based on an invocation or RPC paradigm. Bounded delay is recognised as being the only QoS parameter that has been identified for enabling these applications to describe their requirements. It is the author’s contention that new QoS parameters are required to deal with the characteristics of mobile environments, more

specifically, intermittent and changeable connectivity. Note however, that there may be an overlap between QoS parameters designed for fixed and mobile environments.

The remainder of this section focuses on defining a set of proposed QoS parameters which are aimed to support distributed applications in a mobile environment (additional work is required to establish whether new QoS parameters are required to provide mobility support to continuous media applications, see section 8.4.1). The proposed set of QoS parameters are as follows :-

- available throughput,
- propagation delay,
- idle time, and
- reachability.

Each of these parameters are considered in more detail below through a series of simple examples. The implementation of the new platform's QoS architecture, which underpins these parameters, is described in detail in section 6.4.

6.3.2.2 Available Throughput

The available channel throughput QoS parameter represents the volume of information that can be transferred per unit time (currently in bytes per second) over the current communications medium. To illustrate how this parameter might be used consider the remote database access application described in chapter 5.

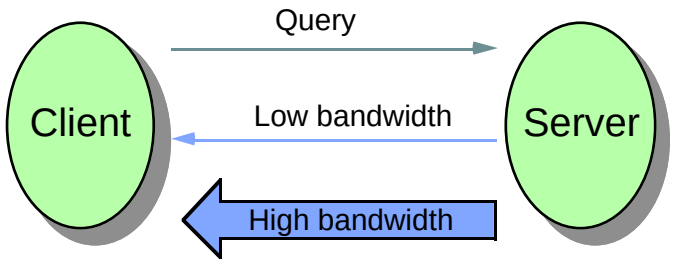

Figure 6.7 - A database interaction

Figure 6.7 illustrates a database access application consisting of two objects: a client who wishes to access the database and a server that provides a public interface on to the database engine itself. The client interrogates the database by sending its query to the server via the platform's invocation mechanism. The server evaluates the query and constructs a found set of database records matching the query. During the client's initial interrogation, the platform is able to calculate the QoS of the communications channel over which the query is taking place. Before the server returns the results to the client, it is able to interrogate the binding between the objects

and obtain an approximation of the throughput. Depending on this approximation, the server can then adjust the quantity of information to return to the client. If the estimate of the throughput falls below some specified threshold, the server could limit the found set to contain only certain key fields, thus enabling the client to further refine its query or request full records of significant keys. If the estimate is in excess of a given threshold, the server could return the entire found set (based on the premise that the performance limit in high speed networks is propagation delay [Grönvall,96]).

6.3.2.3 Propagation Delay

The channel propagation delay QoS parameter provides a measure of the time taken to transmit the first bit of a packet from one end-system to another (in milliseconds). This parameter is particularly significant in conjunction with the available throughput for characterising a given network.

For example, consider the heterogeneous networked environment proposed by the BARWAN project (discussed in section 3.6.4). Their network consists of multiple overlays, from in-building wireless LANs and wide-area packet switched data, to regional area Direct Broadcast Satellite (DBS) and Very Small Aperture Terminal (VSAT) services. This broad range of wireless services will each provide unique service characteristics, both in terms of bandwidth and latency. However, the satellite based services in particular will offer high data rate down-links with especially high propagation delays due to geosynchronous orbit transit times [Katz,96a].

The combination of available throughput and propagation delay parameters would allow an application to apply prefetching and caching tradeoffs. For instance, a file system would be able to prefetch entire file directories to minimise latency in a high bandwidth network. In addition, an application detecting a network with a high bandwidth delay product (such as a satellite based system) would attempt to minimise the fragmentation of its transmission (e.g. by batching messages as much as possible) and use the volume of traffic to amortise the delays incurred.

6.3.2.4 Idle Time

The idle time QoS parameter describes the length of time that is allowed to elapse at an interface without a contact (invocation) from another object before the owner of the interface is notified. The idle time parameter may apply to contact from a specific object or be generalised to detect any contact. The notification (QoS violation) is via a local invocation to a client supplied call-back interface (whose parameters determine the binding which generated the QoS violation and why). The monitoring of idle time may be automatically re-enabled following a notification or cancelled (in which case

the idle time is effectively a one-shot timer). The idle time parameter can be useful in the following generic situations.

Call-back style interactions

An object may request a service from a remote object and supply a call-back interface to receive the results on completion. The idle time parameter would enable the object to detect when results are not forthcoming and take some reparative action (such as trying again, or notifying the user).

Periodic information bulletins

An object may receive periodic information bulletins, perhaps at timely intervals. For instance, the object may monitor fluctuations in the state of the financial markets or collate stock control information to determine purchasing trends. The idle time parameter provides a simple mechanism for discovering breaks in the information supply.

Detecting communications failure

An object may be interacting with a remote object via an unreliable communications link (common in mobile environments). The idle time parameter would allow the object to detect the absence of expected responses, such as replies or service requests, perhaps due to communications failure. For example, typical wide-area communications technologies offer services without guarantees of coverage. In cellular systems there are often coverage blackspots (where no coverage exists), and also, cells may become heavily congested leading to unavailability of service.

A further example of the use of the idle time parameter is provided by the following example (figure 6.8).

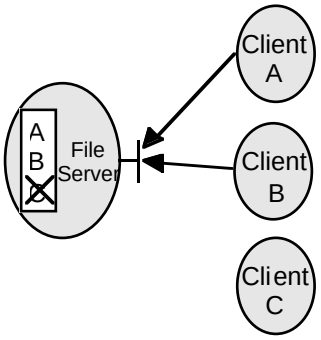

Figure 6.8 - File server object with multiple client objects

Consider a fictitious file system that is constructed using objects. The file server object maintains a service interface to allow clients to request copies of a file. The server would construct a list of clients that were interested in a particular file and endeavour to provide notifications when a client's cached copy was invalidated (similar to call-back breaks in AFS). Each client may periodically ping the server with a 'keep alive' message to inform it that a particular file was still of interest. Using the idle time QoS parameter on its service interface, the file server could easily detect when the keep alive messages stopped being received from a particular client and would consequently be able to remove that client from the notification list for the file.

6.3.2.5 Reachability

The last QoS parameter considered in this section, reachability, is related to idle time, but offers stronger guarantees. Reachability is designed to assert that the absence of a communication from a client within a given threshold is not due to communications failure. The parameter is of particular significance to call-back structured applications. To illustrate why this might be necessary consider the file system example from the preceding section (figure 6.9).

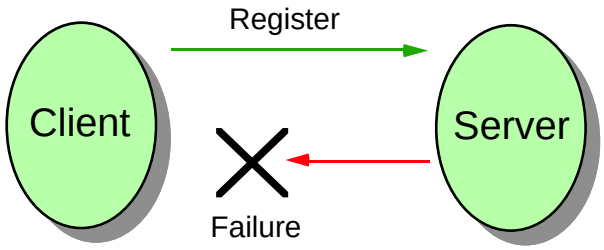

Figure 6.9 - Call-back application structure

As previously stated, a client retrieves some information (a file) from the server and the server warrants to inform the client of any changes thereafter. Therefore, the server is required to keep state concerning which clients it has supplied with which information, together with a call-back interface for invalidation notifications. Structuring an application in this fashion means the client does not have to continually ask the server whether or not the information it holds is still valid (i.e. polling the server). Furthermore, considerable network bandwidth may be saved over the polling strategy; the more files cached by clients, the greater the effective reduction in polling traffic.

Structuring the application in this manner has one significant flaw. Specifically, the absence of change notifications from the server could be for one of two reasons: either the information the client holds is still valid or, alternatively, the communication link is no longer available and so the server has been unable to reach

the client. Optimistically, the client would be using out of date information. However, if the information was of a safety critical nature, the lack of current information is potentially dangerous.

The reachability QoS parameter aims to address this problem by monitoring the availability of the server on the client's behalf, allowing the application to retain its call-back structure. At the lowest level, this maps on to a polling strategy which is controlled by the platform. The main advantage over application level polling, apart from not requiring application restructuring, is that multiple application objects that communicate between the same pairs of hosts can delegate all their polling requirements to the platform. The platform may in turn optimise the actual polling to meet the most stringent polling requirement (by definition meeting less stringent ones) thereby reducing the amount of actual polling traffic required. The platform is also able to use lower level messaging techniques (e.g. the message passing services) that are not available to the application level and incur less communication overhead. Moreover, the platform can obtain information unavailable to an application object to further reduce the necessity for polling, for instance, monitoring interactions between other objects to infer host availability.

6.3.3 QoS Support in ANSAware

The preceding sections have discussed the need for a QoS based architecture and highlighted a set of QoS parameters designed specifically for operation in a heterogeneous networked environment. The ANSAware platform requires extensive modifications, both at the computational and engineering levels, to support the new architecture. More specifically, the following enhancements are required :-

Computational model

The ANSAware platform does not include support for explicit bindings between objects. A new binding architecture is required to enable objects to be bound with an explicit QoS and provide a handle that enables applications to monitor and control the binding.

Engineering model

The current set of ANSAware execution protocols are unsuitable for operation in a heterogeneous networked environment. Moreover, these protocols do not incorporate support for QoS management. A new protocol is required which can operate over multiple network technologies and provide managed QoS information to the binding architecture. In addition, the computational bindings need support from

an engineering infrastructure which enables bindings to be established, controlled and policed.

The following sections consider the computational and engineering QoS support in the new platform in more detail.

6.4 Computational Model

6.4.1 Overall Approach

This section describes the computational features of the new platform for supporting the QoS architecture mentioned throughout the previous section. The extensions to the original ANSAware computational model focus in two main areas :-

i) *Explicit QoS bindings*
The architecture allows applications to abstractly identify interactions between pairs of objects using a QoS managed explicit binding. The binding can be used to establish QoS requirements in terms of the parameters identified in section 6.3 and receive QoS violation notification events, enabling the application to adapt to changes in the QoS provided by the network.

ii) *QoS annotations*
The architecture permits the specification of QoS annotations that can be attributed to specific platform service invocations. Through this mechanism, invocations can have an associated urgency and completion deadline.

These two forms of extension are considered in the remainder of this section. Both extensions required modifications to the platform's DPL language (introduced in section 6.2.5.1). The enhanced compiler grammar describing DPL is presented in appendix A. Finally, the explicit bindings are contrasted with those specified in the RM-ODP.

6.4.2 Explicit QoS Bindings

An explicit QoS binding allows an application object to define an association between a pair of objects. The creator of this binding association receives a handle through which the interactions described by the binding can be monitored and potentially controlled. Information describing the interactions between the bound objects and specifying application requirements are quantified in terms of QoS parameters. In the new platform model, an interface belonging to each party to be associated is required. In addition, one of the interfaces must belong to the object

performing the binding; thus, the owner of the binding is always one of the bound participants. The object instantiating the binding may take the role of either a client or a server, i.e. the binding might describe invocations from the creator of the binding to some remote interface or from a remote client to the creator of the binding. This model is compared with the more familiar RM-ODP model in section 6.4.4.

The creation of an explicit binding can be thought of as comprising of two steps :-

i) The first phase of binding creation is where the binding is initially constructed between a pair of objects (subject to preconditions, such as meeting the appropriate QoS specification). This is the establishment phase. A successful establishment phase will yield a handle onto the binding, enabling the second phase.

ii) The second phase, once the initial binding has been established, is the maintenance phase. During this phase, the binding is being monitored and will report violations in the specified QoS. Furthermore, the handle may be used by the application to interrogate and amend the binding.

Each of these phases is examined more closely in the following sections.

6.4.2.1 Establishing a Binding

A binding is established using a pre-processor directive similar to a trader import or invocation (see table 6.2). Notionally, an object called the *binder* is responsible for checking the establishment preconditions, creating the binding and returning the binding handle (analogous to a binding factory). The binding is effectively a first class object that is responsible for monitoring the binding and may be invoked using the handle (which is simply an interface reference).

In more detail, the binding syntax is expressed as follows :-

```
! {Handle} <- binder$Bind(ClientIR, ServerIR, QoS)
```

A binding may be established by a client or a server object (or both). First, the binder checks to see whether the binding has already been established by another object. If the binding does not exist, one is created with the specified QoS. Once the binding has been created, a local monitoring object is instantiated and an interface reference to the monitoring object returned as the handle to the binding. Consider the example shown in figure 6.10.

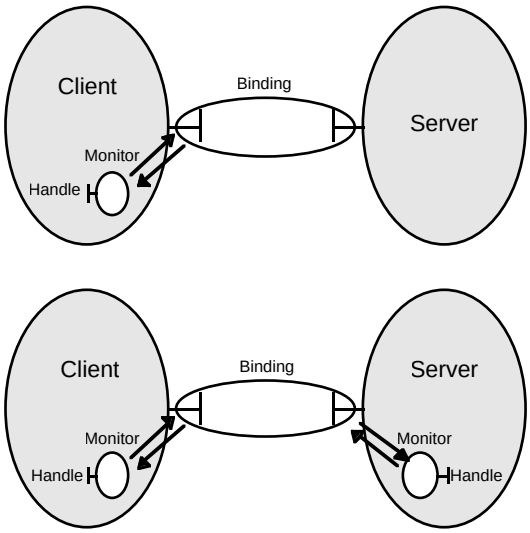

Figure 6.10 - An explicit binding monitored by both parties

The client object executes the binding instructions, indirectly causing the creation of the binding and the instantiation of the local monitoring object. Later, the server executes its binding instructions; the binding has already been established by the client and so only the local monitoring object is established. It is evident that the two objects, the client and server, both have local monitoring objects. Therefore, it is possible that the client and server will obtain different QoS readings for the same binding, according to the perspective of their own monitoring object. This feature, and other differences from more traditional explicit binding approaches, are considered in section 6.4.4. Note that, as previously mentioned, a binding must be established by one of the bound parties; a third party may not establish a binding between two further objects (although, third parties may obtain the binding handle and thereby monitor and control an established binding).

A special case syntax is permitted at server-end bindings, i.e. where ServerIR is a service interface reference that is local to the object creating the binding :-

```
! {Handle} <- binder$Bind(*, ServerIR, QoS)
```

A binding created using this syntax is effectively between any client and the server interface referenced (multipoint-to-point). The QoS parameters supported by these bindings are affected. In particular, the throughput and propagation delay figures will only apply to the first client who contacts the server. In addition, the reachability QoS parameter is not supported for unspecified clients. This mechanism corresponds to a late binding, in which the binding is partially established until contacted by a client. The partial binding can still support the idle time QoS parameter, enabling a server to assert ‘no contact by any client within time *t*’.

The binding control interface that is accessed via the binding handle is considered in more detail in the next section.

6.4.2.2 Feedback and Control

The binding control interface provides the sole mechanism through which a binding is accessed. The binding control interface supports the operations shown in figure 6.11.

```
BindingControl : INTERFACE =
  Unbind : OPERATION [] RETURNS [ Status ];
  GetQoS : OPERATION [] RETURNS [ QoS; Status ];
  SetQoS : OPERATION [ Quality : QoS ] RETURNS [ Status ];
  EstimateTime : OPERATION [ Size : CARDINAL ]
                   RETURNS [ REAL; Status];
  RegisterForEvent : OPERATION [ Event : EventType;
                                 Callback : InterfaceRef ]
                       RETURNS [ Status ];
  DeregisterForEvent : OPERATION [ Event : EventType ]
                         RETURNS [ Status ];
END.
```

Figure 6.11 - IDL specification for binding management

The *Unbind* operation provides a convenient mechanism for closing a binding and at the same time deregistering for all QoS events that have been registered on that binding. The *GetQoS* operation enables the application to query the state of the binding and obtain the current estimates for throughput and delay parameters described earlier. In addition, the information retrieved by the GetQoS operation contains lower layer information about the communications service. This information is supplied by a mobile link manager, called S-UDP (explained further in section 6.5.3.5).

The *SetQoS* operation provides a mechanism through which applications can specify their QoS requirements to the architecture. In the current implementation, the application may adjust a number of execution protocol specific QoS parameters (for example, influencing the RPC retransmission strategies). However, the underlying message passing services do not include QoS reservation transport protocols. Therefore, changes to parameters such as the desired throughput or delay are ineffectual. However, the mechanism could be easily extended to pass this QoS information down to lower layers should QoS reservation support be added at a later date.

The *RegisterForEvent* operation allows an application to register an interest in the occurrence of a particular QoS based event. The QoS events correspond to the set of parameters identified in section 6.3.2. For instance, the application may request to be informed when the available channel throughput exceeds a specified threshold. In this case the corresponding event (*tooFast*) and threshold would be supplied as arguments

to the RegisterForEvent operation. Events corresponding to the throughput, delay, idle time and reachability QoS parameters are defined.

Once an object has registered for a particular event, the underlying protocol takes responsibility for monitoring for the occurrence of that event. Should the event occur, the supplied call-back interface is invoked with an *EventOccurred* operation. The call-back informs the application of the type of event which occurred (separate interfaces could be supplied, in which case the selection of the interface is sufficient to denote the event type). The call-back handler may choose to acknowledge the event and re-trigger the event monitor for a further occurrence of the event or alternatively cause the event be deregistered (in which case the event handler can be considered one-shot). Since the binding control interface is a true first class interface, the object may pass duplicates of the interface to remote objects, allowing them to interrogate the channel QoS or register for events on that binding.

The *DeregisterForEvent* operation informs the binding that the application is no longer interested in events of a specific type. Invoking the deregister operation on a given event is equivalent to the call-back handler requesting deregistration once the event has occurred.

The *EstimateTime* operation provides a convenient mechanism for applications to obtain an estimated total transfer time (in seconds) based on a given quantity of information and the current throughput estimator. The operation relies on a heuristic which calculates the time based on the number of packets the data will require and the amount of protocol overhead predicted. The prediction is effective only as a rough guideline to applications.

6.4.3 QoS Annotations

The extended computational model includes mechanisms for attributing invocations with QoS annotations. The annotations apply only to the single invocation to which they are attributed (maxima or blanket annotations that apply to all invocations on a binding can be specified in the usual way through the binding control interface). Currently two QoS annotations are defined: *deadlines* and *time constraints.*

The deadline† QoS annotation specifies a time within which the invocation should reach the destination service. The deadline affects how the invocation is handled at lower layers (see section 6.5.3.5). An example of the syntax is shown in figure 6.12.

† The platform does not attempt to ensure deadlines are explicitly met by, for example, using real time scheduling algorithms. However, the deadlines do allow invocations to be given relative ordering.

```
! {rarg1, ..., rargN} <- ServiceIR$OpName(arg1, ..., argM) < deadline
```

Figure 6.12 - Deadline specification syntax

The deadline may be specified either as an explicit time in seconds (following the UNIX convention of being the number of seconds since 1 January 1970) or a relative time (specified by the '+' symbol followed by an amount in seconds).

Deadlines are typically used to distinguish invocation traffic of different levels of urgency, for example, giving interactive invocations more significance than periodic background update traffic. One advantage of the deadline scheme, when compared with priority based methods, is the ease with which relative ordering is visualised. However, deadline based mechanisms do have a disadvantage; the closer one is to the expiry of a deadline the more important it seems (short deadlines are associated with urgent invocations). Thus, more recent invocations with high urgency may be passed over in favour of a less urgent one with a deadline that is just about to expire. This dilemma is currently being addressed by the process scheduling community who are aiming to solve the problem by using composite schemes comprising both priorities and deadlines [Nieh,95].

The second QoS annotation gives the ability to place a time constraint on an invocation's execution. If the invocation does not complete (the results have not been received from the server) then a *notWithinTime* exception is generated and the invocation is terminated. The time constraint is set using the *within* keyword and a time in seconds (as shown in figure 6.13).

```
! {rarg1, ..., rargN} <- ServiceIR$OpName(arg1, ..., argM) within time
```

Figure 6.13 - Time constraint specification syntax

A typical example of this parameter in use might be where an object propagates information to a client which is only valid for a certain period. The programmer may put a limit on the time within which the remote object must reply acknowledging receipt of the information. After the specified time, if the information has not been acknowledged, an exception warning the application is generated and the invocation terminated. The application may then take action depending on the implications of the client being in possession of invalid information. The time constraint may be set as a binding parameter in which case it will apply to all invocations over that particular binding.

Both QoS annotations may be used in conjunction, for instance :-

```
! {Result} <- ServerIR$Fetch(Criteria) < +5 within 10
```

This invocation would therefore have a deadline for reaching the server (denoted by ServerIR) in five seconds hence and will terminate and generate an exception if the result has not been forthcoming within ten seconds.

Deadlines and time constraints, as provided by the new platform, are analogous to the soft and hard deadlines specified by the ANSA computational model. However, the QoS annotations in the new platform do not affect the scheduling of application threads in order to satisfy the deadlines or attempt to provide real time guarantees. The implementation of real time constraints has been investigated more thoroughly in real time ANSAware [Li,94].

6.4.4 Analysis

The explicit binding mechanism in the new platform differs from the explicit bindings of the RM-ODP in several important respects.

Binding structure

In RM-ODP, the binding object in a compound binding action provides a central point for monitoring and controlling the binding. In the new platform model, a local monitoring object is instantiated within each capsule that requests to be bound, and this object is responsible for monitoring the binding. To illustrate the point, if two bound RM-ODP objects request the current QoS, both objects are invoking the same binding object and would receive the same answer. In addition, any changes to the binding made by one of the objects will affect the QoS perceived by the other object. In the new model, a binding can be thought of as one object's perspective on the interaction between the pair. This structure is analogous to replicating the binding object locally within each object's capsule, without providing guaranteed consistency. Thus, the QoS requirements at one object are seen as independent from those of the other object.

Third party involvement

A binding object, established by a compound binding action in RM-ODP, is an object which can be instantiated and utilised by a third party. In contrast, the explicit bindings offered by the platform require that a binding is established only by participants in the binding (the binding is internal to these objects). The binding control interface to an established binding may be used by third parties.

Interface causality

In RM-ODP, an explicit binding is made between a pair of interfaces with complementary causalities (ignoring cases involving more than two participants). In the ANSA model, upon which the new platform is based, interfaces are implicitly created with server (consumer) causality. Furthermore, there is no mechanism for creating a client interface with complementary causality (which produces or emits the appropriate operations). As a consequence, a binding must be established between a pair of interfaces with server causality, one from each party.

The first point breaks from the traditional view of an explicit binding, and requires some additional justification. There are three reasons for not supporting the full RM-ODP binding model. Firstly, bindings must be transient over communications failure, as a network incorporating mobile components will have a frequency of disconnection significantly higher than in any fixed network. Secondly, the level of binding management traffic between binding objects required to maintain consistency is undesirable when, for example, a low bandwidth link is being used. Such traffic would be expensive in terms of both cost and time, may require explicit connection establishment and would further consume the scarce network bandwidth. Lastly, not all of the new QoS parameters are orthogonal to both the client and the server ends of the binding. For example, the reachability QoS parameter applies only to service provider interfaces.

6.5 Engineering Model

The computational model described in the preceding section requires a number of supporting components within the engineering framework of the platform. More specifically, these components consist of :-

i) A new RPC protocol, called QEX, which provides all the QoS monitoring functionality required by the computational level binding architecture.

ii) A mobile link manager which multiplexes data across low bandwidth links (i.e. serial and dial-up connections).

iii) Extensions to the UDP message passing service to enable capsule RPC traffic to be indirected via the appropriate transport service.

The relationship between these components and the rest of the platform architecture is illustrated by figure 6.14.

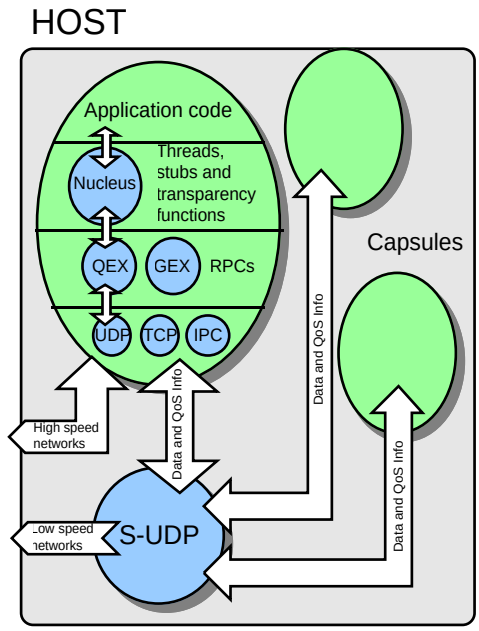

Figure 6.14 - QoS support at the engineering level

As the figure shows, the QEX protocol is a direct replacement for REX. QEX makes use of the modified UDP message passing service to direct traffic via either the network interface (wired connectivity) or the mobile link manager (S-UDP) when mobile.

The remainder of this section discusses the QoS support in the QEX protocol and the mobile link manager in more detail. First, it is necessary to examine REX to establish why it is unsuitable for mobile environments. In addition, detailed knowledge of REX is desirable as a basis for discussing the new protocol, QEX. The section moves on to investigate the backward compatibility issues of QEX and REX.

6.5.1 REX

REX is the execution protocol which underpins the ANSAware invocation mechanism. REX provides remote procedure call semantics [Birrell,84] with streaming extensions to support bulk data delivery [Gifford,88]. The REX protocol has been designed to provide low response times and high reliability for synchronous interactions (interrogations) and high throughput for asynchronous transfers (announcements). End-to-end overheads such as connection management, buffer copying and process scheduling have been minimised. Essentially, the synchronous and asynchronous forms map onto CALLs and CASTs respectively at the protocol level. To summarise the two forms: the CALL (the more commonly used mechanism) is a traditional remote procedure call, i.e. a message invoking a (potentially) remote procedure with a number of typed arguments, followed by the reply containing the response data. A CALL may be either synchronous or asynchronous. In the

synchronous case, the client thread blocks until the invocation reaches a termination. The termination is either the expected reply or an exception indicating failure. In the asynchronous case, the client does not block. Instead, a voucher is returned which may be redeemed at a later date to obtain or block for the termination. The CAST is unidirectional and unreliable: delivery is not guaranteed and no reply is expected.

A REX packet comprises a header portion and a variable amount of payload. The payload component includes sufficient platform information to allow the correct selection of a specified operation on a given interface and a set of marshalled arguments to that operation. A response packet need only contain the marshalled response from the server. In this section we are concerned primarily with the workings of the RPC package and can effectively ignore the payload contents.

The REX protocol makes use of five packet types: CALL, CALLACK, REPLY, REPLYACK and FRAGNACK. All these packet types have a common header structure (shown in figure 6.15).

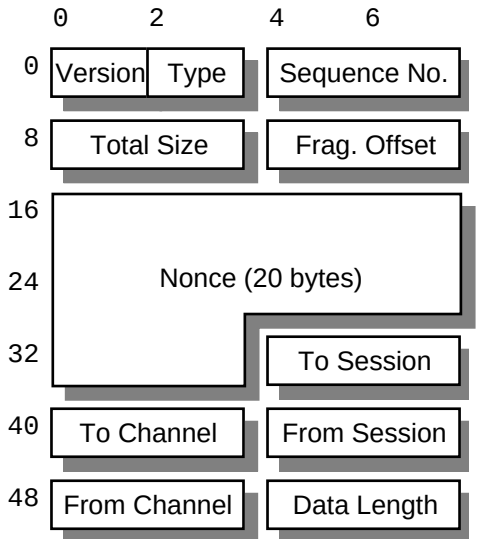

Figure 6.15 - REX packet header

The header is 56 bytes in length and includes fields denoting the protocol version, packet type, sequence number and a unique service identifier (known as the nonce).

The remote procedure call mechanism is capable of passing an arbitrary sized payload between a client and server. Since the majority of protocols place bounds on the maximum size of the packets they may transport, REX operates its own rate-based fragmentation and reassembly strategy. The CAST, CALL and REPLY messages may all be fragmented and reassembled. The exact threshold over which a message becomes fragmented is determined by the underlying transport protocol that REX is using. For example, in the case of UDP this threshold is 1200 bytes, whereas in the

case of TCP, which performs its own fragmentation and places no upper bound on payload size, REX delegates all fragmentation.

The ways in which the various packets interact is illustrated by the following protocol state diagrams (figures 6.16 and 6.17).

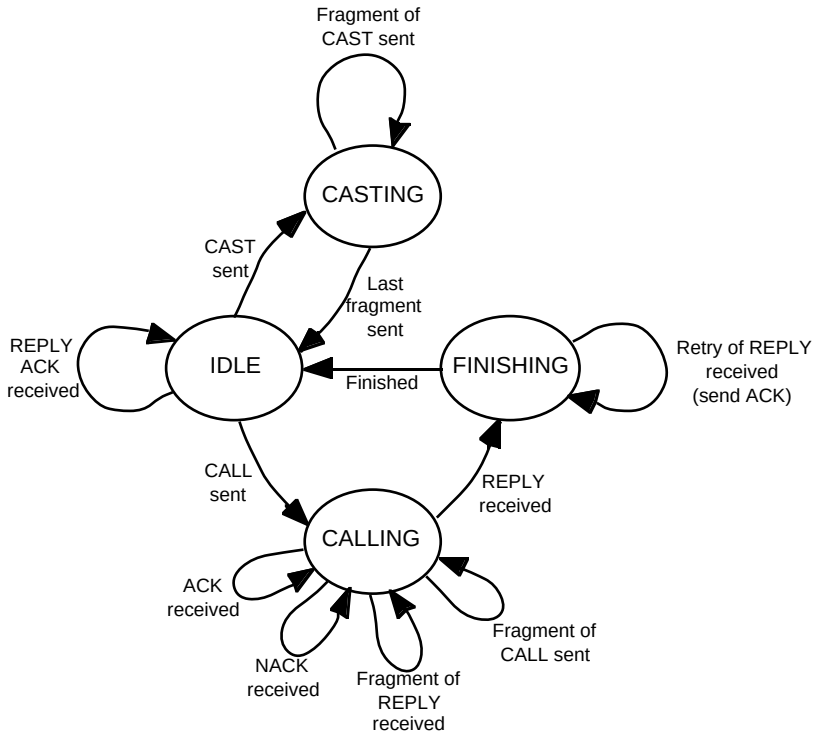

Figure 6.16 - Client protocol states

Figure 6.16 shows the state machine at the client end of an invocation. Initially the client is in the IDLE state. If the client performs an announcement then the protocol transmits the CAST (or fragment thereof) and moves to the CASTING state. The client remains in this state until all fragments of the CAST have been sent before moving back to IDLE. If the client performs a conventional invocation, or CALL, it sends the CALL message (or a CALL fragment) and moves into the CALLING state. After a fixed retry interval, the client will re-transmit the CALL message. This interval increases exponentially unless the client hears from the server or the maximum number of retransmissions has been reached. The client will remain in the CALLING state until either a REPLY message has been received from the server or the interaction times-out. If the server acknowledges the CALL (with a CALLACK), the client moves into a sub-state known as PROBING: instead of full retransmissions it sends only the CALL message headers. These probes occur less often than normal retries and, if not acknowledged, provide an additional method for detecting server failure. Once the REPLY has been received the protocol may move into the FINISHING state. While in this state the protocol will acknowledge (REPLYACK)

any successive retries of the REPLY. Fragmented interactions behave identically with the augmentation that a fixed rate interval timer is used to determine when to send each batch of message fragments.

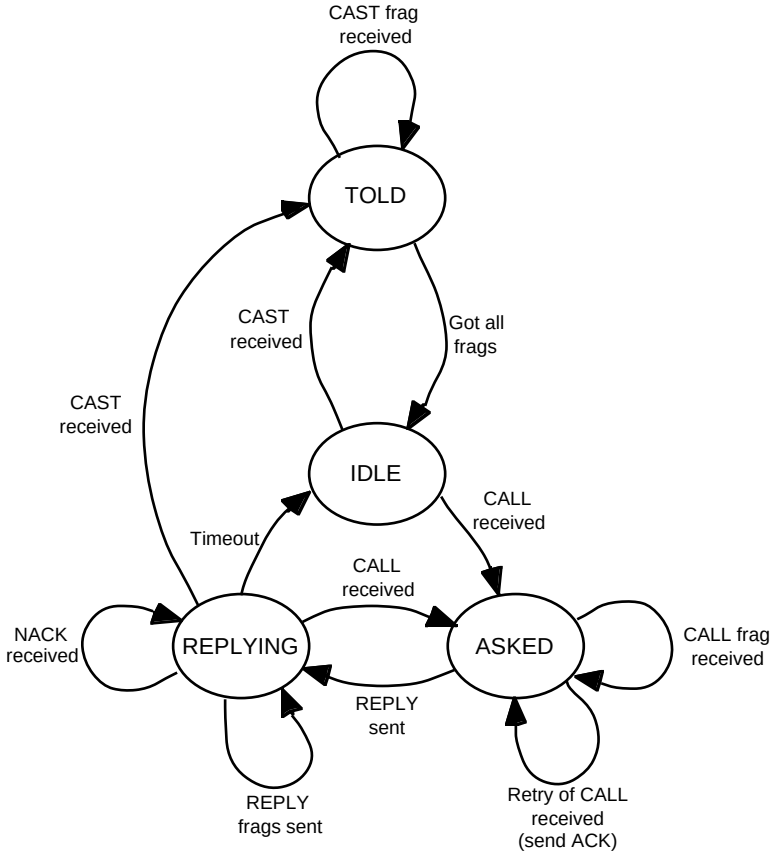

Figure 6.17 - Server protocol states

Figure 6.17 shows the state machine for the corresponding server end of the protocol. As with the client, the server is initially in the IDLE state. Once a message is received it moves into the TOLD or ASKED state depending on whether the message was a CAST or a CALL respectively. The protocol remains in this state until all the fragments (if further are required) are received. If in the TOLD state, once the message is complete the announcement is passed up and the protocol can return to the IDLE state. In the more usual case (from the ASKED state), when the message is complete, the invocation is passed up to the server code. If any successive retries are intercepted the protocol acknowledges them with a CALLACK message. Once the server has composed its response, it moves into the REPLYING state and transmits the REPLY. After a period of time the protocol will drop back to the IDLE state. However, should a CAST or a CALL be received, the protocol will move directly to the appropriate state (TOLD or ASKED) without needing to move through IDLE.

There are some noteworthy features of the REX protocol. Firstly, the protocol makes use of two types of acknowledgement scheme depending on whether or not

invocations fragment. In the non-fragmented case, the first duplicate detected at either end (CALL or REPLY retransmission) is positively acknowledged. In contrast, if the message is fragmented, groups of message fragments (normally 3) are transmitted periodically at fixed intervals. A further fixed interval timer governs when a negative acknowledgement (FRAGNACK) is sent in response. The negative acknowledgement contains a map of fragments the receiving end has yet to see. The map is used by the transmitting end to adjust its map of which fragments to transmit (or re-transmit). Once the completed message is received (all fragments assembled), the entire message is treated as in the non-fragmented case, i.e. if the whole message is a duplicate it is positively acknowledged; if it is a CALL, a REPLY is sent and if a REPLY the next CALL can be sent (if there is one).

Secondly, positive acknowledgements are only sent when a retransmission of a message is detected. A REPLY message implicitly acknowledges a CALL and vice versa, which leads to a special case in which the number of messages to complete an interaction is optimised. In more detail, consider the following example: a client interrogates a server (sends a CALL). If the time taken to service the invocation is sufficiently low and the network fast enough, the REPLY will be received by the client before the client's retransmission threshold is reached and a duplicate CALL is dispatched. The REPLY acknowledges the CALL implicitly and the client may proceed with the next CALL to the server. The symmetrical case also applies. If the duration between the transmission of the REPLY at the server and the reception of the next CALL is less than the server's retransmission threshold then again the acknowledgement message and retry is saved. Thus, during high paced intensive interactions between a client and a server, an invocation requires only two messages (the CALL and REPLY), providing the channel is reliable and none of the messages are lost.

6.5.2 Analysis of REX

The REX protocol has been designed to operate over a particular target network (a moderately loaded wired LAN such as Ethernet). Not surprisingly, when the environment changes, the protocol has not been designed to adapt and consequently ceases to work effectively. There are two specific reasons why REX does not perform well :-

Interval timers

The interval timers in REX, which govern various features such as retransmissions, fragment transmission and negative acknowledgements, have been configured in at compile time. In fact

the three timers mentioned in the previous section, are by default set to 100, 10 and 50 milliseconds respectively. Thus, when REX is introduced to a low bandwidth communications channel (such as those offered by wide-area mobile networks), the fixed interval retry timers are set for too short an interval and the channel becomes highly congested with retransmission traffic.

Congestion control

In order to cope with congestion, the REX protocol makes use of a simple exponential backoff strategy whereby the interval timers are multiplied by a factor based on the number of retries. This strategy is not as effective as a proper congestion control mechanism and recent research suggests that exponential backoff strategies may be inappropriate due to high error rates in mobile environments. This issue is covered in more depth in section 6.5.3.2.

One possible solution, of course, would be to change the default timer settings at compile time and build the protocol to suit the low bandwidth network. While clearly possible, this approach has a number of failings. Firstly, when the channel is used by multiple parties, the effective bandwidth available to the protocol is reduced and the protocol will not adjust satisfactorily to the congestion. As previously mentioned, congestion control is via an exponential backoff strategy. However, backoff strategies cannot replace proper rate control and will inevitably lead to poor and sporadic bandwidth utilisation (the underlying assumption is that congestion is transient, caused by overloading at routers). Secondly, if an alternative network becomes available (such as an Ethernet), the newly tuned protocol will make poor use of the extra bandwidth. Moreover, these problems are further exacerbated if more advanced scenarios are considered (such as those proposed in chapter 2) where a host may be able to make use of seamlessly integrated network technologies with dynamic handoff between different network types.

A further approach would be to use multiple platforms, each tuned for a different network and re-link application code to the appropriate platform when the network characteristics change. However, this approach is clearly inconvenient (and often not possible).

Therefore, a new protocol is required which is designed to adapt to changes in the available network bandwidth. Essentially, the replacement for REX is required to fulfil three major functions :-

i) Underpin the explicit QoS bindings (as described in section 6.4.2).

ii) Adjust to changes in network characteristics as a result of congestion and, significantly, as different network technologies are utilised.

iii) Remain backwardly compatible with legacy ANSAware applications (which may be executing on the local host or made available while connected to the fixed network infrastructure).

In order to remain backwardly compatible, the new protocol is an evolution of REX, retaining many similar mechanisms and using common states, message types and general forms of interaction. The initial engineering problems with QEX were consequently how to adjust the protocol so that it could incorporate congestion control, monitor channel QoS and yet at the same time remain undetectable to existing applications.

Note that, QEX is designed to operate using UDP as the transport protocol. UDP was chosen due to the need to minimise the overhead of the RPC mechanism: the typical RPC carries little data and occurs sporadically, making the overhead of connection-oriented protocols such as TCP undesirable. However, where message size is large and the overhead of datagram headers is significant, TCP may be a more suitable alternative. Clearly, if QEX were to be run over TCP there would be a duplication of functionality. In particular, the congestion control, duplicate detection and packet fragmentation and reassembly mechanisms would all be superfluous. To remove this duplicate functionality, many of the QoS support concepts could be moved into TCP, resulting in a simplification of the RPC mechanism itself. However, following this approach would require changes to the operating system. Currently, no work has been undertaken by the author in implementing QoS support within TCP.

6.5.3 Design of QEX

QEX is the execution protocol which supports the QoS architecture of the new platform. The QEX protocol provides the same semantics as REX. However, the new protocol calculates the QoS of the communications channels using statistics based on packet size and associated round-trip times. The protocol, QoS measurement strategy and backward compatibility issues are described in the following sections. In addition, the facilities provided by the mobile link manager (first mentioned in section 6.4.2.2) are presented.

6.5.3.1 Overview

The approach taken by QEX is to measure the round-trip time (RTT) of pairs of messages sent during interactions to obtain an approximation of the channel

characteristics. This approximation is then used to adjust the protocol (provide congestion control) and underpin the QoS binding architecture.

The QEX state machine was developed using the REX state machine as a starting point (for backward compatibility reasons). To most accurately measure RTT, it is necessary to identify messages that generate immediate responses without incurring additional application delay. For example, if the CALL/REPLY pair were to be used then, once a CALL has been received, there would be an arbitrary delay while the server object services the request (which may potentially involve user interaction), before generating the REPLY message. A complex mechanism would be required to factor out this additional and essentially random component of the RTT estimates.

The most appropriate REX messages, those that are generated without application intervention, are the acknowledgement messages that are transmitted when duplicate messages are received (i.e. the CALL/CALLACK and REPLY/REPLYACK pairs). However, these interactions do not occur sufficiently often (particularly if the REX optimised behaviour occurs) to allow the channel to be continuously monitored.

Consequently, the new protocol generates a positive acknowledgement whenever a CALL or REPLY message is received to enable the RTT to be more easily measured. Additional information (called a tag) is hidden in the header of each outgoing message (the tag structure is described in more detail in section 6.5.3.4). When an incoming CALL or REPLY is received, the tag field is transferred to the corresponding outgoing CALLACK or REPLYACK so that the remote object can easily match pairs of messages.

The QEX RTT mechanism is analogous to the RTT extensions made to TCP for improved performance in high bandwidth networks [Jacobson,92b]. In TCP, a time stamp option field may be added which is echoed back by recipients to enable the sender to calculate the RTT using a single subtraction (additionally, avoiding the aliasing problems identified with earlier TCP window based schemes). In QEX, the time stamps for unacknowledged packets are kept at the sender and indexed using the tag field. On receipt of a positive acknowledgement, the receipt time and tag echoed in the acknowledgement header enables QEX to calculate the round-trip time for that interaction. Tag fields contain an increasing modulo counter which enables packet loss to be detected. QEX is able to perform the calculation using less transmitted information than TCP (the TCP option field is 10 bytes) at the expense of additional processing overhead at the sender.

The tag field is essential for distinguishing retries of individual packets. To explain why this is necessary, consider the scenario illustrated by figure 6.18.

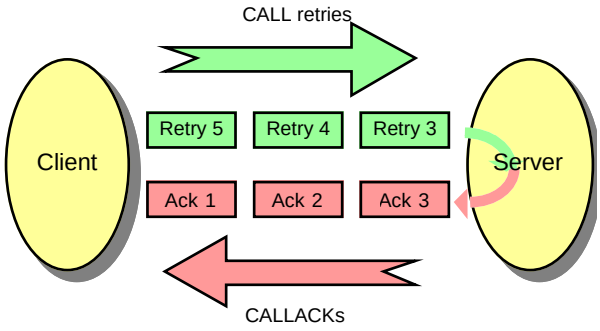

Figure 6.18 - The retry identification problem

The figure shows a pair of objects during an interrogation. The client has begun an invocation to the server and is just about to receive the acknowledgement to the first CALL packet it sent. The client is tuned for a channel which, either through network choice or absence of congestion, was operating faster than the channel it is currently using. Therefore, the client is re-transmitting duplicate CALL packets under the assumption that the prior packets have been lost due to errors, when in fact the first packet has not had time to complete the journey to the server and elicit the response (i.e. the retransmission interval is too short). The server is receiving the CALL packets, transferring the tag from each one to the corresponding acknowledgement CALLACK packet and transmitting it. When the client receives the CALLACK it is able (using the tag) to identify the corresponding CALL and calculate the RTT. The RTT can help the protocol adjust the retransmission timer to avoid unnecessary retries for future invocations.

However, if the tag system is not used, the retransmissions made by the client and their corresponding acknowledgements are all identical. The client is unable to tell which CALL matches which CALLACK. As a consequence the client must assume the acknowledgement it receives partners the retransmission it just made. In the figure this would mean that Ack 1 (the response to the first CALL packet) was paired with Retry 5 (the last CALL retry sent). The effective RTT would make the channel appear to be significantly faster and the protocol would stabilise at this incorrect retransmission rate.

The discussion so far has focused on calculating RTTs for non-fragmented interactions, i.e. where the size of the payload is less than the fragmentation threshold of the transport protocol. In the fragmented case, where the payload is split over a number of messages (fragments) a different mechanism is required. A message pairing is required to allow the RTT to be calculated. By default, the REX protocol transmits bursts of fragments, updating its view of which fragments to transmit in response to received FRAGNACK packets. Each FRAGNACK is generated periodically by a fixed interval timer. Thus, there is no message pairing.

In QEX, it was necessary to make FRAGNACK messages correspond to specific fragments. Contrary to the original scheme which uses a timer to trigger negative acknowledgements at the receiving end, a bit in the flags field of each outgoing fragment triggers the transmission of the FRAGNACK on reception. Negative acknowledgements are therefore driven by the sender in the new scheme. The remainder of the flags field is used as a tag to match acknowledgements to fragments at the sending end (as with the non-fragmented case). The FRAGNACK packets themselves are otherwise identical to their REX counterparts and perform the same function (updating the map of outstanding fragments).

On the surface this scheme seems to advocate a synchronous mechanism where one packet is generated for every one received, which is analogous to a positive acknowledgement strategy. However, QEX contains a slow-start mechanism: in addition to the RTT calculation, QEX calculates the variance of the RTT estimates. The variance is used to determine how stable the current channel estimate is. When the estimate is considered stable (variance lower than a configurable threshold), QEX spaces out the packets which have their negative acknowledgement bit set to reduce the number of FRAGNACKs required. Only when the estimate becomes unstable is the gap between acknowledged packets reduced.

The RTT mechanism, as the basis for adjusting the protocol and gathering QoS information, relies on the underlying assumption that the communications channel is symmetric (offering the same QoS in each direction). In a network where asymmetric channels are possible (e.g. TETRA) this mechanism will be less suitable. In order to work over asymmetric channels, QEX would require a mechanism which calculated the latency. Such a mechanism would require that packets were time stamped and the clocks at each end of the link synchronised (necessitating additional protocol overhead).

6.5.3.2 Backoff Strategies

Each time the retry transmission interval timers are set by the REX protocol, the time interval is increased exponentially. The assumption made by the protocol is that the interval timers are correctly adjusted for the channel conditions and, if no response is forthcoming to a given message, the packet must have been dropped en-route due to overloading at the network routers. The use of exponential backoffs within fixed networked environment is well established and exemplified by TCP [Jacobson,92b]. However, Cáceres [Cáceres,94] identified that in a wireless network (in this particular case a WaveLAN wireless LAN) packets are more frequently lost due to the increased bit error rate and control traffic (such as cell handoff). Cáceres postulates that an aggressive retransmission strategy is more appropriate for such a network (it should

be noted that aggressive retransmission is only applicable once the channel is in a steady state).

In QEX, the RTT variance calculation is used in conjunction with network supplier information to determine the appropriate backoff strategy. Where the bandwidth of the network is sufficiently low and the channel approximation stable, the protocol switches to a linear backoff strategy. The linear strategy leads to more aggressive retransmissions and, ideally, improved performance over error prone links. An exponential strategy is always used in higher bandwidth networks and where a steady RTT has not yet been attained.

However, where a packet must be routed over multiple hops consisting of both fixed and wireless network components, a single backoff strategy may not be appropriate. The most suitable mechanism for the fixed hops would be the exponential strategy, whereas the linear strategy is most likely the best choice for the wireless hops. A possible solution would require RPC agents to be interposed to vary the backoff strategy en-route (an approach endorsed by M-RPC [Bakre,95b]). There is no provision for the insertion of agents in the current QEX protocol. The investigation of improved backoff strategies is the subject of future work (described further in section 8.4.3).

6.5.3.3 Throughput and Delay Estimation

For every remote object, QEX maintains independent channel calculations. These calculations are used to tune the protocol for each interaction. In addition, the calculations support the QoS interface at the binding level (the QoS parameters are outlined earlier in this chapter). Two of the most significant parameters available are channel throughput and propagation delay. Ideally, the throughput and delay components provide an accurate measure of the available channel resources to a given interaction. However, in practice it proves to be far from trivial to convert the available information, such as RTT and packet size, into more meaningful parameters such as these.

QEX currently uses one of two approaches: for want of better names, these are called the simple and complex algorithms (the choice of algorithms is made at compile time to reduce execution overhead). Both algorithms are intended to factor out the two components of channel latency: the bandwidth related component (due to the size of the data) and the delay component (incurred before the data is transmitted). The latter of these two, delay, is the time incurred by the operating system, protocol stack and network interface itself. Traditional approaches rely on measuring sequences of fixed sized packets such as the MTUs in transport protocols (such as

TCP) or the frame sizes of continuous media flows. Since in an RPC the packet size continually varies, these approaches cannot be used.

The simplistic approach is based on the premise that the time taken to send a particular message is directly proportional to the size of the message. Essentially, the delay component is ignored and the throughput must be the size of the packet divided by the RTT. To minimise the affect of variations in the delay component (jitter) the RTT is averaged using a three sample moving average calculation. The number of samples is configurable; the more samples, the smoother the resulting values but the slower the protocol adapts to genuine changes in QoS. In addition to providing a throughput estimator, the smoothed RTT is used to make guesses about the expected transmission time of a packet that is to be sent. The more accurate the guess, the less likely the protocol is to send an unnecessary retransmission. However, if the guess is too large the protocol will be slow to discover lost packets and as a consequence reduce the overall throughput of the RPC.

The complex approach is intended to take into account both the bandwidth and delay related components and, in addition, any sub-packet fragmentation which may occur at the transport layer. Instead of a single calculation, the RTT for each packet size is *plotted* against the size of the packet (within a certain granularity) (see figure 6.19). Calculating the expected transmission time for a given packet size is achieved by looking up the packet size in the graph and reading off the expected RTT (some interpolation between packet sizes may be necessary if the RTT for that particular packet size is not yet known). Determining the throughput of the channel becomes finding the gradient of the line of best fit over the graph. Finding the delay is then the intersection on the time scale of the line of best fit where the packet size is 0. Like the simple approach, individual values of RTTs are smoothed using a moving average calculation.

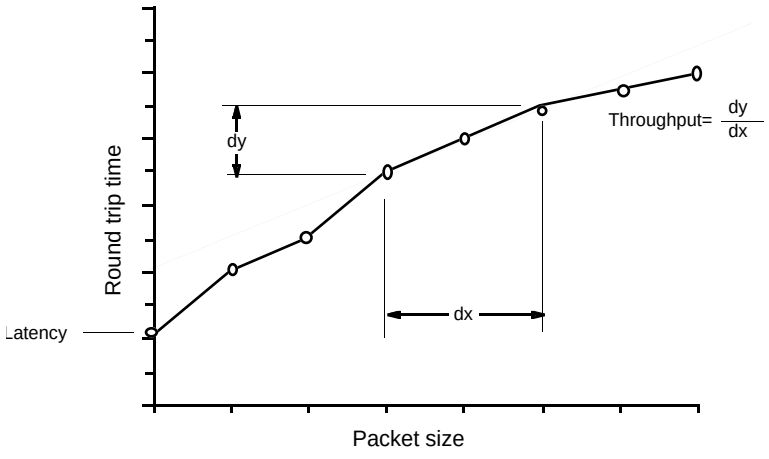

Figure 6.19 - Plot of packet size against round-trip time

This complex approach is computationally far more expensive than the simplistic one. In a bid to decrease the overhead of computation, the information for each interaction is stored in a form of binary search tree known as a red-black tree [Cormen,90]. The red-black tree is approximately balanced: searching and insertion are never more than $O(\log_2 n)$. Searching may be fractionally slower than with a fully balanced tree (red-black trees guarantee that one search path is no more than twice the length of any other), but insertion is considerably less time consuming.

The more complex algorithm has been designed to provide more accurate estimates of RTT by taking into account both the bandwidth and delay components of packet transmission. However, the algorithm has an inherent problem due to way in which RTTs are measured. RTTs are calculated against packet size, so changes in the available bandwidth may be detected by examining how the RTT changes for a given size of packet. Consequently, if the packet sizes vary, it is difficult to determine whether the throughput of the channel has changed. In addition, it is unclear how a detected change for any given sized packet should affect the estimated RTTs for other packet sizes (if indeed it should). To elucidate, if a packet takes time t_1 and a further (but smaller) packet takes time t_2, where $t_2 < t_1$, it is unclear whether t_2 is smaller because the amount of data sent was less or due to a reduced level of network congestion when the smaller packet was transmitted. In the current implementation, same size packets are used to detect changes in the network. The results for other packet sizes are changed proportionally in an attempt to keep RTT approximations more accurate. Observation has shown that in actuality a limited number of different packet sizes are used in most interactions due to the small number of interfaces and operations that are typically used by most applications.

In practice, the simpler algorithm was found to be most effective. This is for two reasons. Firstly, in low bandwidth networks (such as serial lines and dial-up connections), the bulk of the transmission latency is due to packet size, which matches the naive approach. Secondly, in faster networks the variance in RTT caused by factors such as network packet collision and process scheduling delays incurred by the UNIX kernel tend to allow a significant margin of error in the protocol timers. Furthermore, an additional error margin is introduced by the timer granularity (varies between implementations, often in the order of 10ms) and servicing times taken by the UNIX kernel. Clearly, the approach would not fare well in a network which experiences long delays but has plentiful bandwidth (such as a satellite link).

The complex approach, in general, produces more accurate RTT estimates once the channel has reached a stable state. However, up until that point, the protocol is far

slower to adjust in the face of congestion or changes in network (and highly dependent on the type of interactions and sizes of messages used). Furthermore, variance in the RTT experienced by the protocol seriously affects the throughput and delay calculations (the delay is particularly inaccurate). Additional unexpected jitter in wide-area networks can be attributed to transparent lower layer ARQ error recovery strategies.

6.5.3.4 Backward Compatibility

In order to maintain backward compatibility with REX, the QEX protocol has been designed to use packets of an almost identical format. QEX packets can be received by REX objects without being discarded as corrupt (all fields are validated by REX upon reception). The QEX packets contain additional information within the version field (the first two bytes of header). The version field is comprised of two components: a version byte and a flags byte. The version field must remain constant (the incarnation of REX in use with ANSAware version 4.1 is protocol version 3) otherwise REX will again drop the packet. However, the flags field in a QEX packet now contains two new components, collectively called a tag.

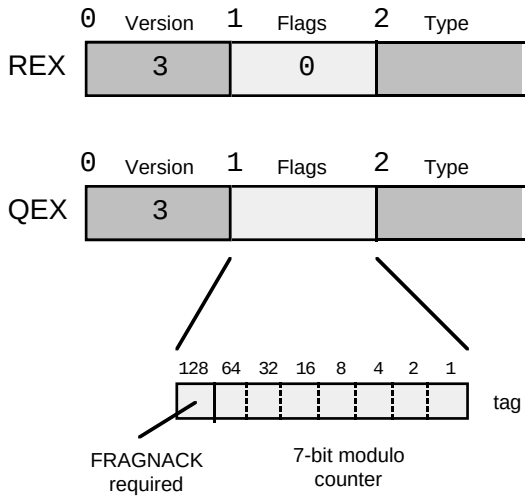

Figure 6.20 - Flag field layout

Figure 6.20 illustrates the first three bytes of the REX and QEX packet headers. The most significant bit of the tag stored in the flags field of the QEX packet holds the negative acknowledgement required bit (which is discussed in section 6.5.3.1). The remaining seven bits hold an increasing non-zero modulo counter. The counter serves two purposes. As the field is non-zero the packet can immediately be identified as a QEX packet (the entire flags byte is always zero in REX). In addition, the counter is incremented and updated in every CALL and REPLY packet that is transmitted by a QEX object. Since the value increases for every message, it is possible to

distinguish between the individual retries of a packet which would otherwise be identical.

A REX capsule intercepting a QEX packet will ignore the contents of the flags field and respond to the packet as if it were a normal REX packet. The REX response will not echo the counter from the received tag, and hence, the QEX capsule will be unable to make RTT calculations (QEX will detect the REX response and move into a backward compatibility mode which is semantically identical to REX).

6.5.3.5 Serial-UDP

Overview

Serial-UDP (S-UDP) is a user level agent process responsible for providing packet transport services over serial line and dial-up connections. S-UDP is analogous to a SLIP or PPP driver. The agent provides UDP-like point-to-point packet transport semantics, i.e. packets are only passed up to the application layer if they are free from corruption and are silently dropped otherwise. In addition, S-UDP provides lightweight support for QoS through which applications or (more usually) platform capsules can receive information about the communications link and exert control over the agent's policies.

The new distributed systems platform requires this additional driver level support for two main reasons. Firstly, the conventional communications API available to applications does not provide sufficient information about, and control of, the underlying communications technologies so that applications can adapt. S-UDP provides a convenient and lightweight method of experimenting with new APIs. A more pragmatic, though equally valid reason, is that bugs within the USL SVR4 kernel of the development platform prevented conventional SLIP/PPP drivers from functioning correctly. Moreover, the lack of available kernel and protocol source code prevented modifications to the existing drivers.

More specifically, the current S-UDP API provides facilities for :-

- Transmission of packets with UDP semantics in deadline order,
- intelligent dialling and hang-up support,
- support for optimising application level QoS parameters and,
- provision of QoS information, such as supplier, charging strategy and monitoring of the delays associated with connection management.

These features of S-UDP are discussed in more detail in the remainder of this section.

Deadline Ordering

S-UDP frames packets with start and end markers, appends a 16-bit checksum and performs any byte stuffing necessary for transmission. The markers enable the remote S-UDP agent to delimit the packets and perform the checksum calculations. The checksum mechanism is based around the CCITT recommended X.25 polynomial $x^{16}+x^{12}+x^5+x^0$ that is also used by some implementations of PPP and is able to detect burst errors of up to 16 bits.

Packets to be sent can have an associated deadline which is passed down from client application capsules. Each deadline has a one second granularity and is specified in terms of the standard UNIX measure of time (the number of seconds since the start of 1970). Packets without specific deadlines are assigned a nominal deadline. The packets are placed in a heap to allow fast deadline ordering and correctly ordered removal. The packets are framed ready for transmission on demand and during idle periods, rather than upon insertion, to help amortise the framing costs (which are linear with packet size). The packets are removed in earliest deadline first (EDF) order and the number of waiting packets and the earliest deadline may be used to implement various connection dial-up policies.

Control Interface

S-UDP supports an interface which client capsules may use to control certain operations and gain information about the network type from the agent. In particular, clients may cause the agent to dial or hang-up the line, register and deregister for events and cause waiting packets to be discarded. The latter option allows capsules to discard buffered RPC interactions to minimise the impact when an RPC is cancelled. A typical use might be to implement an application level cancel operation which requires an RPC to be stopped as quickly as possible (see 5.3.1.3).

Client capsules may register for events such as changes in line state, the absence of a host contact for a specified period of time or the number of messages (and urgency of deadline) waiting in the heap. The line state and heap information events allow for external dial-up and hang-up policies. By default, the S-UDP agent will start dialling as soon as it has information waiting to be transmitted and will hang-up again after a configurable amount of inactivity (by default this is 30 seconds). In a heterogeneous networked environment, the use of multiple wide-area wireless technologies requires that the exact policy will depend on what the user is billed for (per unit data or unit time and so on), the urgency of the traffic and what the user is prepared to pay. The number of options are sufficiently diverse to require either user intervention or that of a sophisticated knowledge base on the user’s behalf. Therefore S-UDP provides support for external decision making policies rather than trying to

engineer a sufficiently general policy into the agent itself. The sequence of messages which enable an application to operate its own dialling policy are shown in figure 6.21.

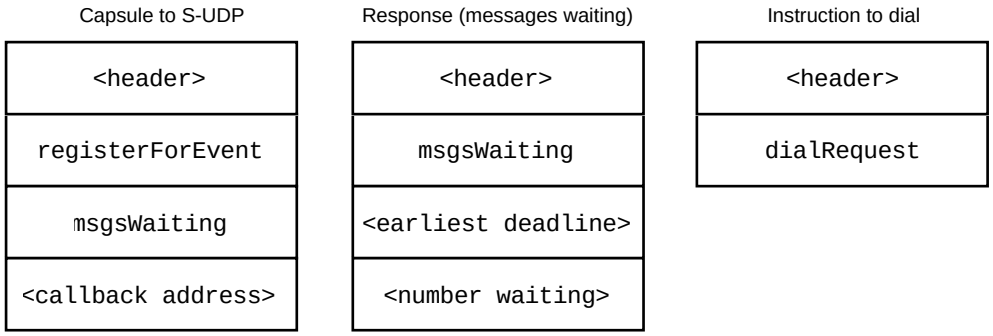

Figure 6.21 - Packet sequence for implementing a dialling policy

The first message registers that the application wishes to be informed when messages are waiting to be transmitted and the line state is down. The response from S-UDP is transmitted when framed packets are ready and waiting and additional packets arrive for transmission (the message includes the deadline of the most urgent message). Based on the deadline, the number of packets waiting and QoS based information (requested separately) the application instructs S-UDP to dial with the last message in the sequence. Messages between the capsule and S-UDP are passed using UDP (the capsule and S-UDP are typically co-located on the same host).

In addition to the *msgsWaiting* event described above, there are a number of additional events. The *hostContact* event has been designed to support the reachability QoS parameter more efficiently across all capsules on a given host. If a client application registers an interest in whether or not a particular host can be reached, the capsule informs the S-UDP agent also. If that particular host communicates with any capsule on the host via S-UDP, all other capsules who register an interest are automatically informed. This saves unnecessary duplication of polling traffic to determine whether or not the host is reachable. This mechanism is transparent to the user.

QoS Support

The control interface allows the client capsules to request the current device QoS (this is done automatically as part of the higher level GetQoS operation on a binding). The QoS parameters retrieved include the network supplier identifier, the dial-up and hang-up delays incurred, the connection and disconnection charges, the charge unit (per byte or per unit time), the cost per unit of information and the current line state. This information is treated as read-only at the application level. All but the dial-up and hang-up delays are currently supplied from a simple database if a particular

supplier identifier is specified on start-up. The dial-up and hang-up times are initialised with defaults from the supplier database but are also monitored over time to improve the accuracy of the estimates.

This QoS information enables applications to employ bandwidth and cost optimisation techniques based on the communications technology being used. Clearly different technologies will require applications to adopt different strategies, particularly where public wide-area technologies are concerned, in order to minimise the cost and delays associated with using the application. For example, a CDPD network would require the application to minimise the information sent by the application, as the charges are accrued according to the quantity of information sent. In addition, CDPD is connectionless and does not require batching strategies to minimise the amount of idle time and amortise the cost and time penalty of connection set-up. In contrast, a connection-oriented service such as GSM incurs charges for connection set-up and for each unit of time the connection is held open. In this case, a strategy would be adopted which made sure enough information was collated before incurring the overhead of setting up the connection. Once the connection was established, however, quiet periods could be used to prefetch information since this additional information incurs no extra cost. Most services apply charges at discrete intervals, per minute for example. Knowing the charge boundary would enable an application to optimise for this threshold, holding open a connection until the next charge boundary to maximise responsiveness without incurring any extra cost.

By supplying applications with managed QoS information the application may be able to adapt and make better use of the supporting technologies, both in terms of performance and increasingly run-time cost effectiveness. It is the author’s hope that, as the concepts of application adaptation are validated, future communications devices and network protocols will provide QoS interfaces to allow each layer of software to best exploit supporting layers.

6.6 Summary

This chapter has proposed that, in order to support complex applications in a heterogeneous networked environment, a greater level of information and control has to be revealed by distributed systems platforms. A new platform, based on ANSAware, has been described in detail. A number of extensions to this platform have been presented which support a QoS architecture based on explicit object bindings. This architecture has throughout been compared and contrasted to the more familiar RM-ODP model.

Current QoS work focuses primarily on support for multimedia data, particularly within the high speed fixed networking community. This chapter has identified a simple set of QoS parameters that are designed to support conventional RPC based applications in a mobile environment. Furthermore, to validate these parameters, they have been implemented in the new platform's QoS architecture. The platform is underpinned by an RPC package called QEX which is essential, both to enable operation in a mobile environment and to gather QoS information to support the binding architecture.

The platform, QoS architecture and QEX protocol are evaluated in the next chapter. In addition, this chapter examines the effectiveness of the platform as a whole in supporting the application prototype described in chapter 5.

Chapter 7

Evaluation

7.1 Introduction

Chapter 5 described an advanced mobile application designed to assist field workers in the utilities industries. The application addresses a number of requirements derived from an analysis of current working practices within the utilities (the industry and working practices are described at length in chapter 4). The application was built on top of a new distributed systems platform based on ANSAware which has been explained in some depth throughout the preceding chapter.

The platform is intended to provide an environment for constructing adaptive mobile applications: that is, applications that are aware of their mobility and are able to adapt their behaviour in response to changes in their environments. It is the role of the platform to provide a QoS architecture which allows managed information from lower layers to reach interested applications and offer them a greater degree of control over their supporting infrastructure.

In this chapter the application and platform are evaluated. More specifically, section 7.2 discusses the application in terms of how well it was perceived to meet the end-user requirements established by the MOST project. Section 7.3 considers the platform's role in supporting this application in a mobile environment. Section 7.4 focuses on adaptation within the application and the platform itself. Finally, a performance evaluation of the QEX protocol is presented in section 7.5.

7.2 Evaluating the Application

7.2.1 Involvement of End-users

The application prototype has benefited from the involvement of an end-user organisation† from the earliest inception of the project. The ongoing requirements capture process, and particularly the initial field study, was conducted within their organisation. Subsequently, during the design and implementation of the application prototype, the end-users were involved in an evolutionary process of meetings, demonstrations and critical feedback. This process enabled the application to more accurately reflect the wishes and needs of the industry.

The process culminated in an evaluation of the concepts embodied by the application through a limited field trial. Selected members of the end-user organisation enacted a specially designed scenario using the application prototype. The scenario was chosen to as accurately as possible reflect the real life situations faced by the utilities, within the constraints of the non-disclosure agreement in force. All information exchanged during the scenario was fabricated and any real schematic or geographic information was suitably sanitised for publication. The scenario was chosen carefully so that these constraints did not affect the evaluation process. However, as a consequence the scenario did not exercise the full range of the application's functionality.

The feedback from the end-user organisation as a result of this liaison and, in particular, following on from the field trial are presented in the next section. The field trial scenario is discussed in more detail below.

Scenario: *The trial scenario consisted of an emergency unscheduled work item caused as the result of a road traffic accident. The accident and the potential damage caused by the car is reported to the fault report line of the utility company by a police officer who attended the scene of the accident. The utility company representatives were required to perform the necessary operations to effect a repair using the prototype system. The steps taken are laid out in the following dialogue.*

Office Perform a database query to locate the information pertaining to the damaged network section. The resulting information consists of two records: a geographic record containing a map which fixes the position

† As previously stated, the name of the company cannot be revealed for confidentiality reasons.

of the fault to a specific wiring pit and a schematic record which details the network switching and connectivity of the wiring in the pit.

Establish an audio and GIS dialogue with a field engineer (audio was via a separate mobile phone).

Use the GIS module to remotely display the geographic record retrieved from the database and mark the location of the pit using the highlighting (or *red-lining*) tools. Answer any queries about the pit location collaboratively.

Fill out an electronic job instruction instructing the field engineer about the nature of the problem in order to be able to officially turn over control of the damaged section of network. Forward copies of the instruction to both the field engineer and the local records office to issue a permanent record. The field engineer is then cleared to proceed with the investigation of the scene of the accident in more detail.

Field Travel to the scene of the accident and analyse the damage.

One of the vehicles has been partially pushed into the pit. The fuel tank has been ruptured and there is a risk of explosion.

Report the status of the situation back to the office and request the necessary assistance to expedite the job.

Office From the pit schematic record, determine the most suitable feeders that must be isolated in order to repair the damaged pit whilst maintaining the maximum amount of connectivity to other sections of the network.

Use the collaborative GIS to coordinate the switching of three substations in order to complete the necessary isolation of the damaged pit.

Formulate a repair strategy which includes the necessary switching, supply restoration and earthing procedures. During the switching procedure an up-to-date picture of the current network state is disseminated to all parties and updated as switching operations occur using the highlighting tools of the GIS module.

Field Fill out electronic job completion forms and attach the pit schematics that have been updated during the repair. Send the record to the office for inclusion in the site safety document.

7.2.2 Feedback from End-users

The end-user organisation, both during the development process and following the field trial, identified a number of interesting issues which could form the basis of future work on the application. These comments were of particular interest to the project partners, EA Technology, who intended to take elements of the application forward into product quality software.

7.2.2.1 Group Representation and Selection

The central panel of the group coordinator user interface (see figure 5.6) illustrates the users currently in collaboration by representing each participant with a digitised photograph. Arranged in a column under each participant is a list of icons representing the modules currently in use by that user. Each column of modules is arranged so that the same module for each user aligns to form rows. Group operations from a particular module are only propagated to those modules in the display which have been selected (this process was previously described in section 5.3.1.3).

This mechanism was judged to be an effective and powerful method for controlling the interactions within a collaborating group. However, the interface is somewhat overwhelming for inexperienced users. Consequently, extensions would be required to provide novice and expert modes to the interface: the novice mode would propagate all group operations to all member modules, and the expert mode would use the current interface.

7.2.2.2 Look and Feel

As detailed in chapter 5, the application was structured as a toolbox of sub-components or modules. Each module conformed to a simple and flexible modular design which permitted the application to be extended to include new functionality as easily as possible. This structuring technique proved to be effective both from an implementer and a users' point of view, allowing user requirements to be quickly reflected in the application. However, two recommendations were put forward for future development following the end-user's use of the prototype.

i) The first suggestion was for a consistent look and feel across all the application modules. For example, if input is required from the user the same mechanism should be used to commit the entered text (pressing return or clicking a button). In a collaborative project such as MOST, implementation of this recommendation would have required adherence to a well defined user interface style guide. A style guide was not developed due to time constraints

placed on the development process and the evolutionary nature of the prototype.

ii) The second suggestion was to enforce a more prescriptive mechanism for moving the user from module to module. For example, on receiving a job instruction, the job dispatch module should be started automatically. If a diagram is received with the job instruction, the GIS module should start up displaying the attached information, and so on.

7.2.2.3 GIS Requirements

Initial versions of the prototype allowed for dynamic creation and destruction of GIS monitors or views. Users could create a view of a particular map or schematic, capture the operations and paste them into other views. In addition, the user could designate which of the available views were to propagate operations to other group members.

This mechanism was abandoned in favour of a more structured interface consisting of a single public and a single private monitor. The new mechanism was found to be far simpler to use due to the clear distinction between propagated and non-propagated operations.

Once the system was in use for any period of time, it became clear that the highlighting mechanisms needed two further extensions to maintain usability. The first was the requirement for additional labels to highlighting marks. When a number of similar highlighting marks were in use on the same monitor it became quite difficult to distinguish which of the marks were the significant ones without a frame of reference. This requirement was partially satisfied by allowing the colour of each highlight to be selected. Further extensions to highlight significant marks or pop-up labels would enhance the usability of the system.

The second highlighting tool extension required was for supporting transient highlights or gestures. These new highlighting marks would not be permanently attributed to a particular map or monitor, but simply attract the users attention to a particular area. Supported gestures would include pointing to a particular location, circling a location and tracing a specified route on the display.

7.2.2.4 Reflection of Mobility

In a mobile environment, particularly where low bandwidth networks are employed, remote operations or operations which propagate to remote users often introduce delays into the application. Conventional user interface design is largely based on the premise that an application is local to the user and so the response time

to any given operation is small. This assumption reflects on the type of components available to the application designer. For example, having clicked upon a button on the user interface, the user is given very little feedback about how the operation is progressing which often leads to frustration; often causing the user to click on the button multiple times (analogous to pushing a lift call button repeatedly because the lift has not yet arrived). More informative displays are required which provide feedback about the ongoing state of the operation. One such example might be a button which has ‘idle’, ‘operation underway’ and ‘operation completed’ states (though additional states such as ‘contacting host’, ‘trying again’ and so on may also be useful).

7.3 Qualitative Platform Analysis

In project terms, that is with respect to the user requirements, the platform can be deemed successful. In particular, it enabled the application prototype to operate in a variety of networked environments and, significantly, underpinned the field trial discussed in the preceding section. This section examines the computational and engineering models of the platform and comments on the appropriateness of the abstractions and supporting mechanisms it provides. This section is based primarily on the experiences, problems and enhancements highlighted during the development of the prototype and, as such, contains no quantitative results (these are considered in section 7.5).

7.3.1 Computational Model

The explicit binding extensions provided by the new platform enabled the development team to construct a variety of applications. Of these, a number make use of the new facilities to adapt their behaviour in response to changes detected by the supporting infrastructure (these are covered in more detail in section 7.4). The binding syntax was found to be a familiar and natural extension to use, primarily because of its similarity to the existing invocation syntax. Moreover, using the binding control interface was no different to using any other object within the environment. However, there were a few observations worthy of note :-

i) Forming a binding between a local client and a remote server object was quite straightforward, since a server interface must be obtained by the client object anyway before invocations can begin. However, the explicit binding operation requires interfaces to identify both parties (see section 6.4.2). The client typically uses its own management interface (from which all interfaces descend) to identify itself in these binding actions. Obtaining this interface

reference was found to be non-trivial, requiring a client to create a service interface and invoke a management operation. A more convenient mechanism was required. The binder was extended to support the `Bind_GetLocalMgmt` function to simplify this process.

ii) The establishment of a binding between a local server and a remote client object was found to be less natural, as the programming paradigm does not usually involve managing interfaces which relate to clients. In practice, this was rarely a problem; in situations where a server was interested in the QoS between itself and the client, an interface reference was already known (for either call-back or the provision of reciprocal services). In addition, in some cases, the server could avoid contacting the client for an address by using the special case binding syntax (any client to single server).

iii) The binding operation takes a QoS structure as the last parameter which specifies application QoS requirements to the binder. This could be tedious as the programmer must provide default values for each field within the structure. To assist the programmer, an additional binder function `Bind_InitialiseQoS` was provided. This function initialises the structure passed to it with a suitable set of default values. Using this function, the programmer need only adjust the fields that are of interest before creating the binding.

iv) The throughput and delay QoS parameters are useful for making decisions concerning the volume of information to transport. However, it is often difficult to realise how these parameters map on to a transmission time because of the lower level issues, such as fragmentation and protocol overhead. Since the application level is unaware of the exact transmission scheme used (and this level of detail should remain abstracted) a new operation was created to help. The *EstimateTime* operation (see section 6.4.2.2) makes an approximation based on the channel characteristics and lower level protocol knowledge about the time taken to transfer a specified volume of information. The resulting estimate is approximate and does not explicitly take into account retransmissions due to errors.

7.3.2 Engineering Model

As previously discussed, the platform's QoS architecture consists of two engineering components: the binder and the QEX protocol. This proved to be a successful paradigm; meeting the requirement for interoperability and enabling both conventional and mobility-aware applications to operate in a range of networking

environments. This section examines the outstanding issues that were raised during the development and use of the existing platform implementation. The performance aspects of QEX protocol are considered in section 7.5.

i) The QEX protocol has been designed to use the UDP message passing service (see section 6.5.2). Conventionally, the UDP protocol operations (e.g. `sendto`) will block during underlying dial-up connection management. However, in the current platform architecture, the capsule sends to the S-UDP agent (section 6.5.3.5) which queues the packets and transmits them when the link is available. This process has the unfortunate side-effect that the S-UDP process is always available and thus, while the link is down, the capsule may continue sending packets to S-UDP. These packets are most commonly retransmissions of earlier packets (as no response has been forthcoming) and hence represent unnecessary or redundant traffic. Furthermore, where dial-up delay is long (perhaps due to connection failure), higher level invocations may time-out before connection is established, but the queued packets will still be transmitted. The current solution to this problem is for each capsule to be a registered client to the S-UDP QoS interface. The S-UDP agent informs these clients as to the state of the interface (link up or down etc.) when a change occurs, enabling capsules to adjust retransmission timers to include or exclude dial-up delays (as necessary). This solution has the additional advantage that the application is not blocked while the communications link is down.

ii) Protocol interfaces which buffer packets before transmission (in particular S-UDP) suffer from an additional problem which is emphasised by connection-oriented links. If at a higher level the operation generating the traffic (for instance, an interrogation) is invalidated, either through a time-out, failure to meet timeliness QoS or a user cancel action, the packets relating to that operation are still queued for transmission. Once the link has been established, these redundant queued packets will waste valuable bandwidth (and time) unnecessarily. The current, rather inelegant but expedient solution, is for the capsule to inform the S-UDP agent that the packets are invalidated and S-UDP to apply platform specific knowledge to identify the packets to discard. Clearly, platform level knowledge should not be required at the S-UDP level. An alternative solution would involve tagging packets of a common interaction with a channel identifier and supplying this identifier to S-UDP with the cancel instruction.

iii) Currently, the architecture does not provide applications with a convenient mechanism for controlling the selection of a networking technology on a per interaction basis. Ideally, an application would be able to select which bearer technology was used based on requirements such as the urgency of the interaction, associated cost and network criteria discussed earlier. The logical choice for interfacing to this functionality would be through the QoS parameters passed to the Bind operation. However, the current system makes use of a low level function call which controls the destination of all the traffic from the capsule.

iv) The accuracy of the channel characteristics approximations that are calculated within each capsule (see section 6.5.3.3) are affected by their operating environment. More specifically, running within user space on non-real time operating systems such as UNIX, introduces random jitter due to lower level scheduling functions. A more accurate approximation could be obtained by moving the monitoring functionality closer to the network (ideally into the network protocols themselves) where knowledge about the total traffic to and from a host can be applied. This information could then be supplied to the capsule as QoS information.

v) The current architecture does not provide a mobile-aware group RPC mechanism. The platform does, however, inherit the ANSAware group RPC mechanism known as GEX. GEX was *not* used to support the application prototype for three reasons. Firstly, the protocol employs a highly synchronous token passing algorithm which requires rapid communication of intensive peer-to-peer messages, impracticable within most mobile environments. Secondly, the protocol is layered on top of the unicast RPC protocol and would, assuming a suitable network, represent little advantage in terms of performance over using the RPC directly. Lastly, the protocol offers a fully transparent transactional service in which a group invocation either succeeds upon reception by all participants or fails atomically. Considering this last point in more detail. In a failure prone environment, it may be desirable to weaken the transactional semantics to allow successful delivery to subsets of the multicast group. A QoS framework for tackling this issue is being developed [Cheverst,96]. More specifically, the framework will provide mechanisms for discussing application data consistency, timeliness and cost requirements within a group, enabling protocols to apply tradeoffs and optimisation techniques to maximise useful work under such conditions. In addition, the QoS feedback concerning group member consistency is expected to enable applications to adapt to cope with the semantic change.

Moreover, through adaptation and lower level optimisations (such as use of multicast protocols and network multicast support) the framework is anticipated to increase group performance in mobile environments.

The next section looks in more detail at the role of adaptation within the application prototype and the supporting platform itself.

7.4 Analysis of Adaptation

The central argument in this thesis is that applications are required to adapt to changes in their underlying supporting infrastructure in order to operate effectively in heterogeneous environments. Specifically, since applications are responsible for generating the bulk of communications traffic, application specific knowledge is required in order that intelligent action may be taken in response to environmental changes. Furthermore, lower level support for adaptation is necessary to enable the monitoring of the environment, collation of information and provision of managed information up to the application. In the following sections, the ways in which the application prototype and the supporting distributed systems platform are able to adapt is evaluated. A range of techniques which can be applied in adaptive systems are also highlighted.

7.4.1 Application Level

The prototype application has been designed to operate in a heterogeneous networked environment. More specifically, the prototype is able to make use of a range of networking technologies including fixed networks (Ethernet) as well as analogue (AMPS/TACS) and digital cellular telephones (GSM). The prototype is required to interact with remote users via this diverse infrastructure and cope with the range of QoS offered by these technologies. In the following sections the modules within the application that are most affected by the need to adapt to these changes in QoS are presented.

7.4.1.1 Remote Database Access

The remote database access module, introduced in section 5.3.4.1, was required to demonstrate that a mobile field worker could access a remote database server running within the fixed office based infrastructure. The requirements analysis process identified that the application should permit the field worker to be able to obtain a list of search topics, search based on record type, retrieve result lists of matching records and create, edit and delete records (subject to access permissions) based on these results [Worship,94].

In the utilities industries, a large number of independent database systems are employed during everyday operations, these include: customer records systems, plant location and stock control, fault report and analysis systems, job management systems and, increasingly, on-line geographic and schematic databases. For the purposes of the field trial, access to a mock-up system containing dummy customer records was sufficient to demonstrate that the prototype was capable of interworking with remote systems. The mock-up database contained no sensitive information.

The content of the customer database chosen for the field trial was based on analysis of an actual customer database subsystem. The customer records were required to store a range of information including the following fields (those fields that should be indexed by the database to permit searching are marked with an asterisk (*)) :-

- customer surname (*)
- customer initials
- customer title
- customer supply address
- customer supply postcode (*)
- customer telephone number at supply address
- billing address and telephone number (if different from supply address)
- tariff code
- billing frequency
- meter type (prepayment or credit) (*)
- peak rate meter number
- off-peak rate meter number
- last two peak rate meter readings
- last two meter readings (estimated or actual)
- number of appliances on maintenance contract with company
- details of each appliance serviced by the utility (including manufacturer (*), model (*), year of manufacture, start date of maintenance agreement and duration of the agreement)
- presence of other fuels (such as gas, LPG, oil etc.)

In a real system this information would be cross referenced against the composite mains record system to enable the location of the particular customer and schematic details of their supply.

The volume of information returned by a database depends on the generality of the query (how many matches are found) and the size of the records to be retrieved. The implications of the size of this 'found set' depends on the underlying communications technology in use. For example, fetching 1K of information at 9.6Kbps would take approximately 1.2 seconds to transfer the data (including the protocol header but excluding connection set-up time, the time taken to transfer the query and process it at the server). A found set of 1Mb under the same conditions would take over 20 minutes. Using a wired 10Mbps Ethernet the same 1Mb of data would take just 1.08 seconds (under ideal conditions).

Clearly the volume of information must be minimised when using low bandwidth communications links in order to maintain usability. The amount of reduction necessary is dependent on the volume of information and the data rate of the link. The remote database access tool is the only point at which there is sufficient application specific knowledge to intelligently make such a tradeoff. For instance, reducing the quantity of information to essential fields (such as the searchable fields) over a low bandwidth network. In the prototype application this decision is simplified by the limited amount of information stored in the mock-up database. The tradeoff is made based on the platform's approximation of the current throughput and the number of records matching the query. The information is made available by using the EstimateTime operation of the binding interface described in section 6.4.2.2.

In terms of the field trial scenario, the field engineer was required to perform database accesses to determine the location of the damaged pit and retrieve appropriate geographic and schematic information. If, for argument's sake, the database record had a more up-to-date schematic state diagram attached (schematics range from a few thousand bytes to several mega-bytes depending on the size of area covered and whether the image is vector or raster based), the implications of pulling the entire record would have a serious impact on the system performance. Instead, when the engineer is browsing the database, the system is able to apply the bandwidth tradeoffs and over a low bandwidth link return a subset of the information to enable the engineer to refine the query or request the remaining information. Over a high bandwidth link there is often little point in optimising the amount of data returned as the biggest performance bottleneck is in terms of the latency of performing the requests. In the prototype application the record number, customer name, initials, title, meter type and address fields represent the minimal amount of information returned.

This cut down record form takes only a few seconds over a typical low bandwidth link but provides sufficient information to help refine the query to determine if more information is required.

7.4.1.2 Lightning Warning Service

The lightning warning service, first introduced in section 5.3.4.4, provides a value added service enabling clients to access a centralised lightning information repository and receive early warning of an impending electrical storm. The lightning warning system is structured as an event based architecture. The client registers the user's location (a postcode is sufficiently accurate) with the central authority which will then call-back the user in the event that lightning strikes are detected in neighbouring areas. However, operation in a mobile environment raises an interesting issue, illustrated by figure 7.1.

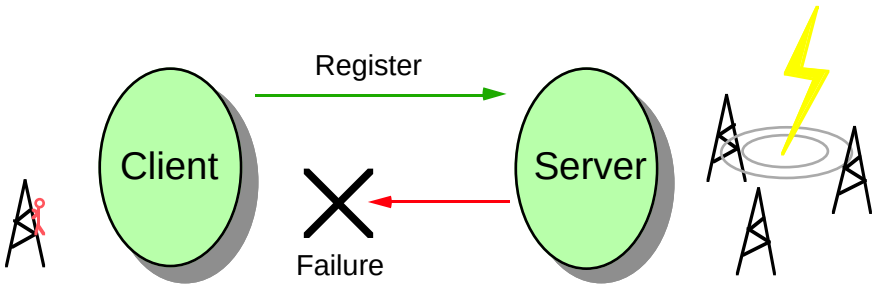

Figure 7.1 - Lightning warning service call-back structure

As discussed in section 6.3.2.5, wide-area wireless technologies are often characterised by intermittent connectivity due to interference, communication blackspots (where no coverage exists) and oversubscription within the cell. The lightning warning service is consequently unable to determine whether the absence of information from the repository is due to the lack of danger or the server's inability to contact the client. To address this issue the service uses the platform's explicit binding mechanism to establish a binding between itself and the repository. The service is then able to request a suitable reachability QoS parameter, delegating the responsibility to the platform for ensuring the repository is reachable and therefore able to deliver suitable warning messages. If the service is found to be unavailable, the QoS violation call-back from the platform enables the application to warn the user that the lightning service is unavailable and that appropriate action should be taken.

The platform allows the application to maintain a natural event based programming model without resorting to application level polling. Furthermore, the platform polling necessary to maintain the assertion is undertaken and optimised

transparently, in many cases being obviated by observing low level traffic between objects or by the characteristics of the communications technology being employed.

7.4.1.3 Mode of Working

Interestingly, the group configuration tool influences the working patterns of field engineers (the group coordinator module is discussed in section 5.3.1.3). In particular, the user may implicitly change between synchronous and asynchronous working to increase the performance of collaborations. As previously discussed, the application prototype supports facilities for synchronous collaboration between field engineers using spatially referenced data. The engineer is able to browse locally stored maps and schematics and perform a number of highlighting operations upon them either locally or in a shared workspace. Shared operations are propagated synchronously to selected group members.

In addition to this synchronous mode of working, operations can be packaged up into clipboards which may be attached to structured E-mail messages for asynchronous propagation. The engineer is consequently able to choose between synchronous and asynchronous message interactions with colleagues based on a variety of requirements such as message urgency, cost and so on.

A given instance of the application may be interacting with other application instances via a variety of bearer technologies. Since each technology will have a particular QoS, the time taken to propagate each operation will vary depending on the particular technology used to communicate with each party. Consider an end-user of the application interacting with an office based colleague via a fixed network. If the user should add a mobile field engineer to the conference they will notice that the pace of the collaboration drops considerably. The operations which are propagated to the mobile field engineer will suffer from the additional delays associated with the communications technology, typically low data rates, increased error rates and dial-up delays.

However, if the user is made aware of the implications of communicating information to the various conference participants, they can choose between synchronous and asynchronous interaction styles. Indeed, as the quality and cost of the communication varies, users respond by adjusting their behaviour between synchronous and asynchronous working [Cheverst,96]. In the example scenario above, the user might choose to continue the interaction collaboratively with the office based colleague and then periodically send updates asynchronously to the field based engineer.

To facilitate this behaviour, the current prototype provides feedback on the underlying communications through the central panel of the group coordinator module. During collaborative interactions the central panel displays a horizontal row of icons representing the other participants of the conference. Explicit bindings are used to enable the monitoring of the network QoS to each party. Based on QoS information, the group coordinator changes the background colour of each user icon to represent the relative connectivity of the group members. A green background to a user's icon implies a healthy connection to that user, whereas a red background implies a relatively poor one (analogous to the status lights of many PMR radio handsets). So, in the example scenario, the office based colleague would be shown on a green background and the field based colleague on a red background representing that the bandwidth available to the office based user was higher than that to the mobile user.

7.4.2 System level

The preceding sections have highlighted the role of adaptation within the modules of the application prototype. These adaptive modules enable the application to operate in a heterogeneous networked environment. Adaptation is also a requirement within the supporting platform in order to cope with the variety of network QoS offered by the different communications technologies. This section discusses how adaptation within the application and platform itself is enabled by the supporting infrastructure. The discussion considers three aspects of adaptation in turn: monitoring, protocol adaptation and feedback.

Monitoring It is essential that the characteristics of, and significantly the changes in, network QoS are monitored. The accuracy of the channel measurements are paramount, since the remainder of the architecture, both in performance and adaptation terms, rely upon them. However, measurement of the channel characteristics are non-trivial. This issue is covered in more detail in section 6.5.3.3.

Adaptation The primary use of the channel information is in enabling protocol adaptation. This level of adaptation is critical to maximise the system performance given the available network resources. The current protocol adapts in two ways. Firstly, the retransmission strategies are tuned to minimise the amount of bandwidth wasted by unnecessarily retransmitted packets. The reduction in wasted bandwidth decreases the time taken to complete an interaction. More specifically, it minimises the number of packets required, reduces the delays caused

by the queuing of valid packets behind unnecessary ones onto the network and, in addition, speeds error recovery as lost packets are detected more quickly. Furthermore, the increase in available bandwidth permits a higher degree of channel sharing (more concurrent interactions) given the same maximum available bandwidth. The second adaptation technique concerns the backoff strategy, which is adjusted to aid the performance in error prone channels. Essentially, a linear strategy can be adopted which aims to avoid the misinterpretation of bit errors as network congestion. There are however a range of associated issues which have already been highlighted as requiring future work. The real time performance of the protocol's adaptation is considered further in section 7.5.1.

Feedback The feedback of channel information to higher layers is at least as crucial as protocol adaptation. Feedback is the critical process that enables application adaptation. However, the interface and form of the information is also highly significant. For instance, if the information collated by the protocol were supplied in raw form to the application, the frequency of the interactions would severely impact on the system performance. More importantly however, the type of information (packet round-trip times) would be almost meaningless to the application without the additional contextual information that is available at the protocol. Therefore, the platform is required to present the information in a form that is intelligible to the application and, in addition, does not in itself impact extensively on the system's performance. The current strategy addresses these requirements in two ways. Firstly, the application chooses the information that is appropriate through the binding control interface. This means that the information is both relevant to the application and managed by the platform to achieve the desired form. Secondly, the feedback from the platform is in response to changes in the observed behaviour of only the specified fields and, additionally, only if the fields transgress the bounds of acceptability chosen by the application. The change event call-back system requires far less interaction with the application than an approach based on polling (which enables the application to observe the changes for itself).

There are a number of additional lower-level forms of adaptation which can improve both the system performance and, potentially, the cost effectiveness of the whole application. For instance, there are a wide range of optimisations that can be

implemented based on the type of communications technology in use. By adapting the system when a change of technology occurs, to optimise to the characteristics of a given technology, performance can be improved. In the current system, the S-UDP connection manager and MPS services implement a batching strategy for connection-oriented links to reduce the number of explicit connections that must be established. Furthermore, packets waiting for transmission may have an associated deadline. Packets are transmitted in earliest deadline first (EDF) order. These strategies can be controlled via a QoS interface to the connection manager which enables more complex optimisation techniques to be implemented. In addition, QoS information denoting the characteristics of the communications technology is integrated with the QoS interface provided by the platform to enable higher level operating decisions to be made.

7.4.3 General Discussion

The work of this thesis has shown the importance of adaptivity and how it applies at all levels of a system. A number of useful adaptive techniques have also been highlighted. Some of these techniques have been adopted by the application and platform (as presented in the preceding sections). Table 7.1 summarises the most significant techniques for adaptation highlighted by this thesis. The table is for illustration purposes and is not intended to present an exhaustive summary.

Level	Technique	Description
User	Change of working practices	The user can alleviate demands on the network, e.g. change task, swap from synchronous to asynchronous collaboration or specify which tasks are most important to them.
Application	Restructure using agents or delegation of processing	Processing/network intensive tasks can be offloaded to remote sites or pre-processing or filtering applied to remote data (reducing bandwidth requirements and freeing host for other tasks/dozing to save power).
	Use proxy services	The application can use local substitute services based on cached information (often with reduced functionality) while disconnected.
	Change model of interaction	Interactions can be adjusted from polling to event based structures or from RPC to an alternative (perhaps asynchronous) paradigm.
	Reorganise application structure	One example of application restructuring is to change from using distributed state to a centralised architecture to simplify consistency management in unreliable conditions.
	Re-bind to new services	The application may be able to rebind to equivalent services which are easier/cheaper to access. Alternatively, it may be possible to migrate the service or application component.
	Change application demands	New QoS requirements can be negotiated or non-essential bindings dropped. Alternatives may be possible, e.g. lossy encoding.
	Adjust consistency requirements	Groups may be able to tolerate weaker consistency or adjust operations to achieve quorum, yet avoid hard to reach members.
Middleware	On-demand cache management	Information can be fetched only when needed, instead of speculatively, e.g. opening the first page of a document and transferring successive pages later, or retrieving e-mail headers before message bodies.
	Prefetching into the cache	The application can fetch information while the link is good, in case it is required when the link degrades or becomes expensive.
	Apply filtering and compression	The volume of information to transfer can be reduced by compression or filtering non-essential frames from hierarchically encoded data.
	Efficient protocol utilisation of the channel	The transport mechanisms can be adjusted to match channel characteristics, e.g. retransmission/backoff strategies, header compression, error control and handling of asymmetric channels.
Transport and below	Change or introduce new protocols	New protocols can be selected which suit the characteristics of a particular network or appropriate protocols can be introduced (e.g. injecting a reliable data link layer).
	Optimise data for the network	Protocols can adjust their packet sizes to suit different networks. The operating system can adjust the queue sizes onto the network interfaces which impacts on latency, particularly of multimedia streams.
	Optimisation of multicast	Multicasts can be mapped onto the network technology, particularly those with partial or full hardware multicast support.
	Optimise for the characteristics of the network	There are a number of cost and network structure optimisations. For instance, batching data to spread the dialling delays, or transferring additional information while the time is already paid for.
	Reordering of data	The priority or urgency of data may require that it is handled preferentially in scarce bandwidth situations.
	Demultiplexing to multiple networks	If multiple technologies are available simultaneously, it may be advantageous to use several at once.

Table 7.1 - Summary of common adaptation techniques

Note that, although an attempt has been made to categorise these techniques into various system layers, a number of techniques are applicable throughout a system's

architecture. For instance, caching can clearly be applied at a variety of levels; obvious examples include world wide web browsers which cache at the application level, whereas the caching performed by a file system would be at a lower level. In addition, efficient channel utilisation is as much an issue within platform level protocols (such as RPC mechanisms) as it is within lower level transport protocols (such as TCP). The range of adaptive techniques which are applied within a system will depend as much on the requirements of the application and the types of data it uses, as the performance characteristics of the networks that are available.

7.5 Quantitative Platform Analysis

The distributed systems platform is required to provide mechanisms by which applications can specify their requirements in terms of desired QoS and receive violation notifications should these requirements be unattainable. In order to provide such mechanisms, the platform must gather and manage the QoS which, in turn, places a certain performance and communication overhead on the platform's layers and, more specifically, the QEX protocol.

In the following sections, the overhead of gathering the QoS information is examined in performance terms by comparing the QEX protocol used in the new platform with the original protocol used by ANSAware, REX. As previously stated, the QoS information is also used to enable the protocol to adapt to the characteristics of a given channel. The performance of this protocol adaptation process is considered in the following section, before looking at the wider implications on protocol performance.

7.5.1 Rate of Adaptation

The QEX protocol (see section 6.5.3 for a full description) is required to adapt to changes in network QoS in order to avoid causing, and be able to react to, congestion. Importantly, in a heterogeneous networked environment, the protocol is also required to detect and adapt to the dramatic changes in QoS caused by a switch in supporting network technology. More specifically, the QEX protocol, must adjust its interval timers to avoid wasting bandwidth with unnecessary retransmissions. Furthermore, this transition must occur as quickly as possible to maintain the accuracy of the RTT calculations which supply QoS information to the binder (spurious QoS information may induce bogus QoS violations, causing the application to adapt unnecessarily). However, this requirement for fast adaptation must be balanced against the need to remain as impervious as possible to fluctuations caused by jitter.

The rate of adaptation of the current QEX implementation is illustrated by the following figures. Figure 7.2 illustrates the number of unnecessary retransmissions that are sent before the protocol adapts to the new bandwidth. The rate change axis of the graph depicts the fractional change in bandwidth (for example, a fractional change of 1/2 implies that the bandwidth has dropped to half its original value). The figure is calculated by averaging over different bandwidth ranges (i.e. 9.6-4.8Kbits/sec, 4.8-2.4Kbits/sec, etc.). The largest drop in rate shown is where QEX adapts from the default interval values coded into REX to a 300bps channel.

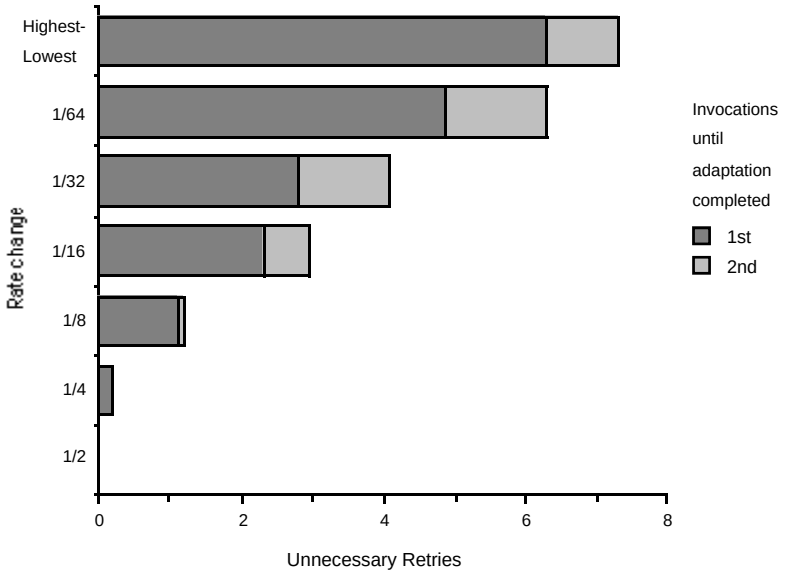

Figure 7.2 - Unnecessary retransmissions per invocation until adaptation completed

In the absence of a suitable heterogeneous network test bed, the results shown in the graph were taken using a software network emulator developed at Lancaster University [Davies,95a]. The network emulator offers an approximation of the QoS experienced by applications over a variety of channel characteristics including shared channels. Furthermore, the emulator can switch between network configurations instantaneously under user direction to simulate the transition between seamlessly interconnected network technologies. The change in rate is instantaneous (in a real system some handover latency may be experienced, this issue is being addressed by the BARWAN project, see section 3.6.4). No bit errors or packet losses were emulated during the tests performed on the protocol.

The results were collated from a simple client and server, the client messaging the server using non-fragmenting invocations. The size of the outgoing CALL packet (including platform headers but excluding UDP and IP headers) was 215 bytes. The corresponding REPLY packet totalled 140 bytes. Test statistics were averaged over three independent test runs for each bandwidth transition. Once a network transition is triggered, the protocol will continue transmitting at the stabilised rate until a change in

round-trip time is detected. As soon as the protocol is aware of the change in channel characteristics, it begins to adapt to the new rate and so fewer unnecessary packets will be transmitted for the next invocation, and so on.

The horizontal bars in the graph illustrate the total number of unnecessary retries that were sent while QEX is adapting. This total is composed of the unnecessary retries sent during the first two invocations following the change in rate (for the changes tested, no more than two invocations are needed before the protocol has fully adapted). In particular, where bandwidth halves, no unnecessary retries occur in adapting to the new rate. Where the bandwidth drops to an eighth of the former rate, on average one unnecessary retry is sent during the first invocation. Over the maximum drop in bandwidth no more than 8 unnecessary retries are sent with approximately six of those occurring in the first invocation.

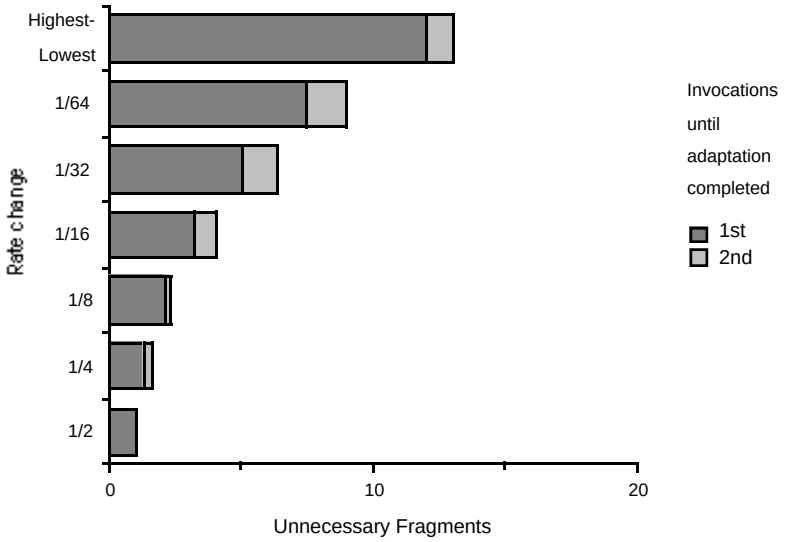

Figure 7.3 - Unnecessary fragments transmitted per invocation until adaptation completed

Figure 7.3 portrays the same testing process with fragmented invocations. The payload carried in each invocation is 2048 bytes, which fragments into two CALL packets of 1378 and 1026 bytes respectively. As with non-fragmented adaptation, the protocol continues to transmit at the stabilised rate until a change in round-trip time is detected. However, in the fragmented case, on discovery of the rate change the protocol requests explicit negative acknowledgements after every fragment until it has sufficient round-trip times to adapt such that no redundant fragments are transmitted. Upon stabilisation, the tight fragment/negative acknowledgement coupling is relaxed over time.

In the non-fragmented case, unnecessary retries are completely redundant and constitute a waste of valuable network bandwidth. In the fragmented case however, extra fragments transmitted while the protocol is adapting are only redundant if they

are retransmissions of fragments that the remote object has already received. If the change in bandwidth occurs during a long invocation (with many fragments) then these fragments would have to be transmitted anyway and can be treated as congestion with the usual backoff strategy. As with non-fragmented invocations, the protocol needs no more than two invocations to complete adaptation in all the test cases.

The QEX protocol always starts off with the same set of defaults as the REX protocol. To minimise adaptation at start-up time, every capsule updates a common file on each host referred to as an *experiences file*. The experiences file contains the stabilised protocol parameters keyed from the QoS information presented to it from S-UDP. Therefore, for every independent technology (or *supplier*) reported by S-UDP, defaults are available which are likely to more closely match the channel characteristics than the protocol defaults. S-UDP is not used during the testing sequence and so the experiences file will not influence the test results presented throughout this chapter.

7.5.2 Evaluation of General Protocol Performance

The preceding section dealt with one specific feature of the QEX protocol, i.e. how it adapts. The following sections turn to analysing more general aspects of the protocol's performance. The analysis is broken down into three test cases, dealing with non-fragmented interactions, non-fragmented interactions in error prone channels and fragmented interactions separately. The results presented throughout the following sections were captured using the network emulator mentioned earlier.

As previously discussed in section 6.5.3.1, QEX calculates message round-trip times by positively acknowledging CALL and REPLY messages with CALLACK and REPLYACK messages respectively. These acknowledgements are sent immediately upon reception of the generating message before the data is passed up to the application to avoid introducing unpredictable application delays into the RTT calculations. The overhead of calculating these round-trip times is analysed in the following sections.

7.5.2.1 Test Case 1 : Non-fragmenting Invocations

The first set of results presented (figure 7.4), compare the number of invocations per second attained for the QEX and REX protocols over a range of emulated network bandwidths. The figures were taken using the same client and server test pair mentioned in section 7.5.1. A test run consists of ten non-fragmenting invocations, each corresponding to an outgoing CALL message size of 215 bytes and

corresponding REPLY of 140 bytes. The figures presented are averaged over three independent test runs for each bandwidth considered.

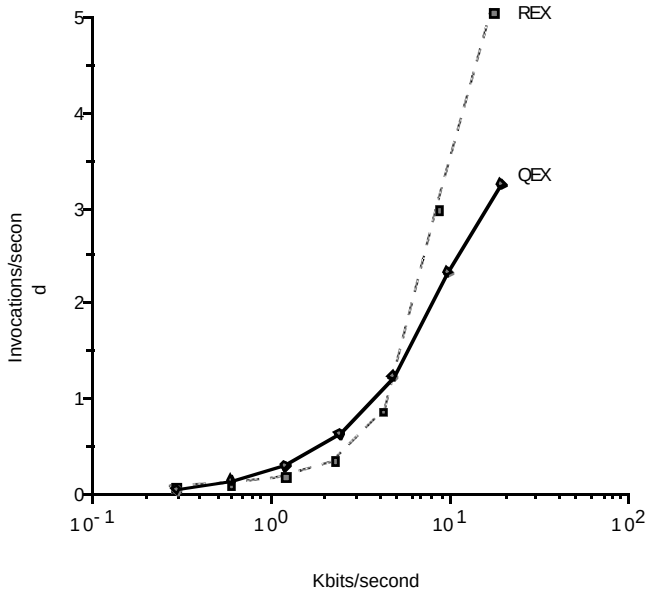

Figure 7.4 - Comparison of the number of invocations per second attained against bandwidth

As can be seen from figure 7.4, below an emulated bandwidth of around 9.6Kbps the QEX protocol permits a higher number of invocations per second. Beyond this threshold the REX protocol is faster. This behaviour can be explained by examining the test scenario in detail (the REX protocol and this optimised behaviour is described more thoroughly in section 6.5.1) :-

- The test client issues an invocation to the test server (a CALL message) which upon reception at the server illicits some computationally inexpensive service.
- The response from the server (a REPLY) is dispatched following completion of this request.
- Once the test client is in receipt of this response it may begin the successive invocation (the REPLY acts as an implicit acknowledgement for a CALL and vice versa).
- If this REPLY has been received before the retransmission interval of the client, then a retry of the CALL has been saved. The converse case, where the successive CALL implicitly acknowledges receipt of the REPLY, also applies.

Therefore in the test case, where the test server is performing a simple service and the network bandwidth is sufficiently high (from around 9.6Kbps) the REX optimised behaviour is occurring and hence fewer messages per invocation are being sent (note that QEX is unable to use this optimised behaviour because of the positive

acknowledgements; it must send at least the four CALL, CALLACK, REPLY and REPLYACK messages in order to work out the channel RTTs).

Despite the overhead of RTT calculation, the QEX protocol does have a number of benefits, particularly at low bandwidths (see figure 7.5).

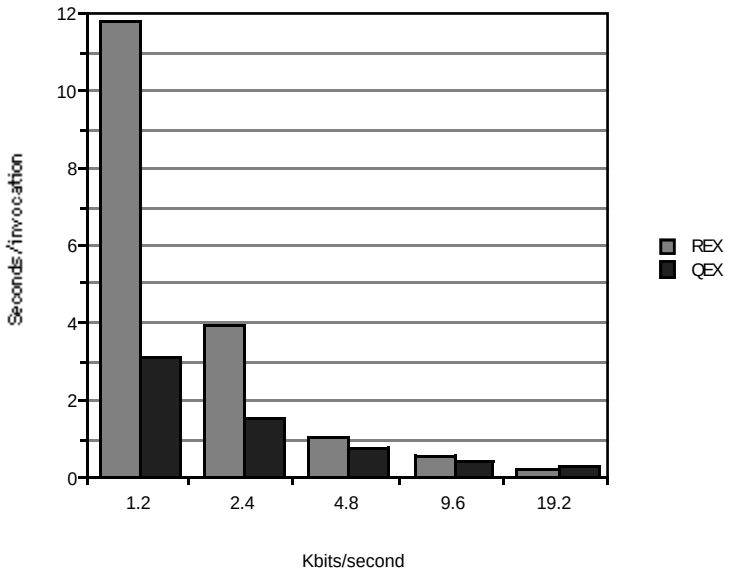

Figure 7.5 - Comparison of the number of seconds taken per invocation against bandwidth

Figure 7.5 shows an alternative plot of the same measurements presented in figure 7.4. This new plot illustrates that at an emulated bandwidth of 1.2Kbps a QEX invocation is approximately four times faster than using REX. Furthermore, at 300bps (not shown in this graph) a QEX invocation takes approximately 15 seconds, against over 80 seconds with REX. However, as the bandwidth increases to over 9.6Kbps, the optimised behaviour of REX starts becoming more apparent and eventually REX surpasses QEX in terms of performance. The reason behind these figures is more clearly illustrated by examining the number of packets that are sent in order to complete each invocation (figure 7.6).

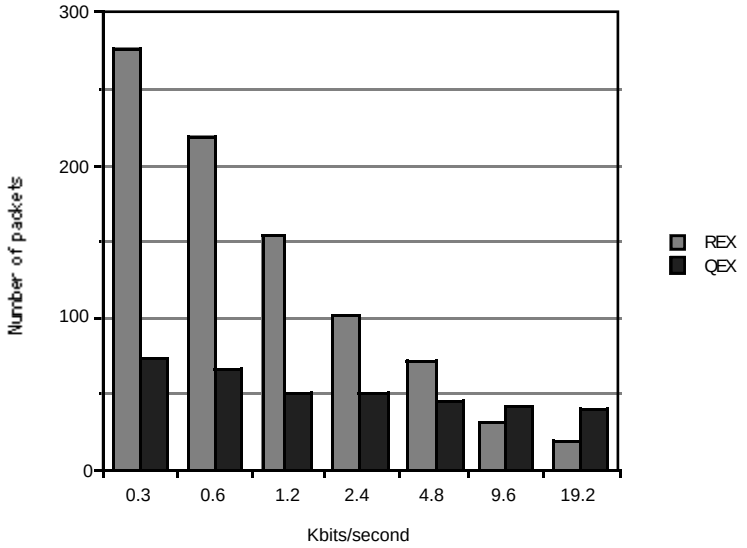

Figure 7.6 - Comparison of the number of packets sent during a test run against bandwidth

The figure shows the number of packets used to complete a test run of ten invocations. The congestion problems introduced by using REX at low bandwidths, despite the in-built backoff strategy, can be clearly seen. The number of packets used by QEX to transmit the required information is fairly constant across the range of bandwidths (as one would expect). The amount of unnecessary retries translates to a lower percentage of user data being passed by the protocol in each invocation (as illustrated by figure 7.7).

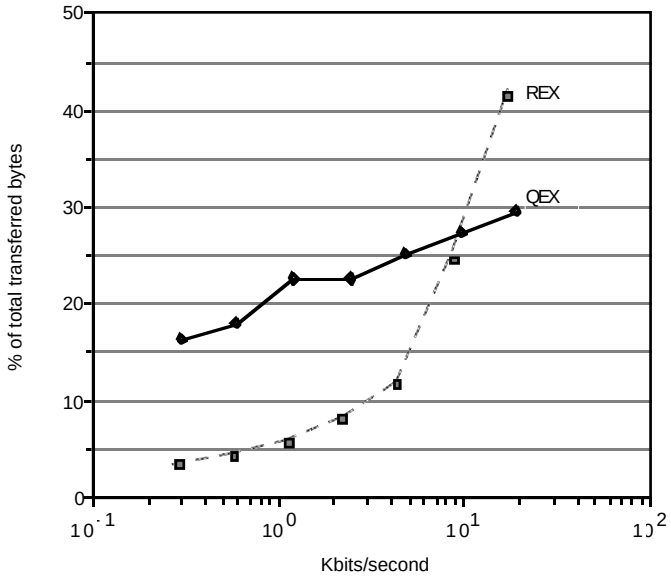

Figure 7.7 - Comparison of the percentage of user data transferred per invocation against bandwidth

The figure confirms the expectation that, the more unnecessary packets are used, the lower the percentage of user data that is transferred. The graph implies that REX

follows an exponential curve, transmitting more and more user data with respect to bandwidth. However, figures indicate that both QEX and REX stabilise at their respective number of packets minima and, although invocations are faster at higher bandwidths, the percentage of user data transferred remains constant (since REX will transmit fewer packets in the optimised case the percentage of user data is constantly around 10% higher than with QEX).

Overall, the percentage of user data transmitted is never more than around 40% due to the overheads introduced by the protocol packet and ANSAware headers (clearly this figure is dependent on the ratio of packet payload against header size). The REX (and QEX) header is currently 56 bytes with an additional nucleus overhead of 44 bytes (this last factor varies depending on the interface operation used). Thus, excluding overhead of the protocol stack underpinning ANSAware, the minimum header size is 100 bytes. To address this issue, additional work has been carried out to produce a version of QEX which incorporates header compression along the lines of that proposed by Jacobson [Jacobson,90]. The revised protocol is known as CQEX [Hirschmann,96].

7.5.2.2 Test Case 2 : Non-fragmented Interactions with Emulated Packet Loss

The second test case illustrates the effect of jitter and packet loss on the protocols. The client and server test pair, as presented above, were both configured to introduce random processing delays of up to three seconds before the start of each invocation and during request servicing at the client and server end respectively. The effects due to the delays introduced at the client were removed from the invocation timing statistics. The random delays introduce jitter into the RTT calculations as the platform does not have a pre-emptive thread scheduler and so cannot interrupt the processing during delay periods to service packets that are waiting for attention. The number of invocations in a test run and the sizes of the packets used are identical to those used in the previous test cases.

In addition, the emulator was configured to provide a channel between the object pair which corresponds to a wireless bus, that is, a single common channel. Contention to transmit on the channel was emulated, which results in packet loss if both objects transmit at the same time. The channel supported by the emulator is analogous to an open channel PMR system. However, packet loss due to bit error rates caused by channel effects was not emulated.

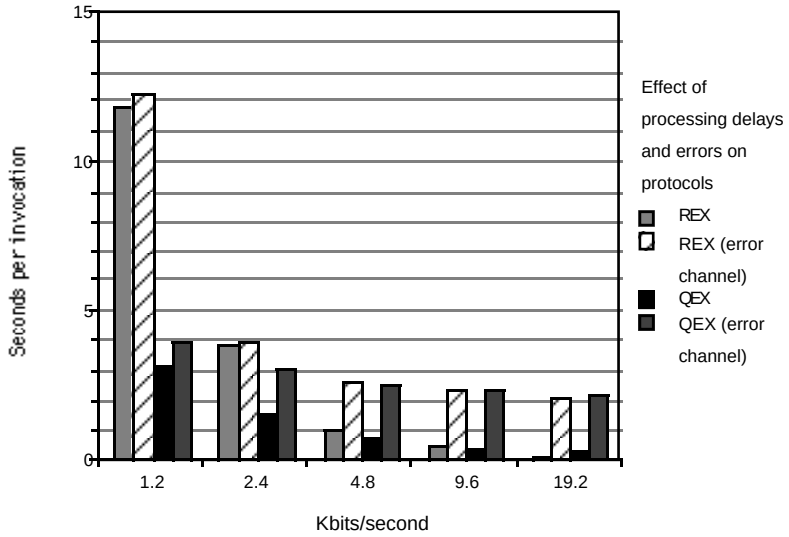

Figure 7.8 - Comparison of the effect of errors and delays on the protocols

Figure 7.8 shows the net effect of the jitter and packet loss on the protocols with respect to the test case one scenario. The additional delay in servicing the invocations can be clearly seen. Indeed, the delay becomes the dominating factor from data rates of 4.8Kbps upwards. However, packet loss will account for a percentage of the additional delay seen for each invocation. The impact upon the protocol can be more clearly seen by considering figure 7.9.

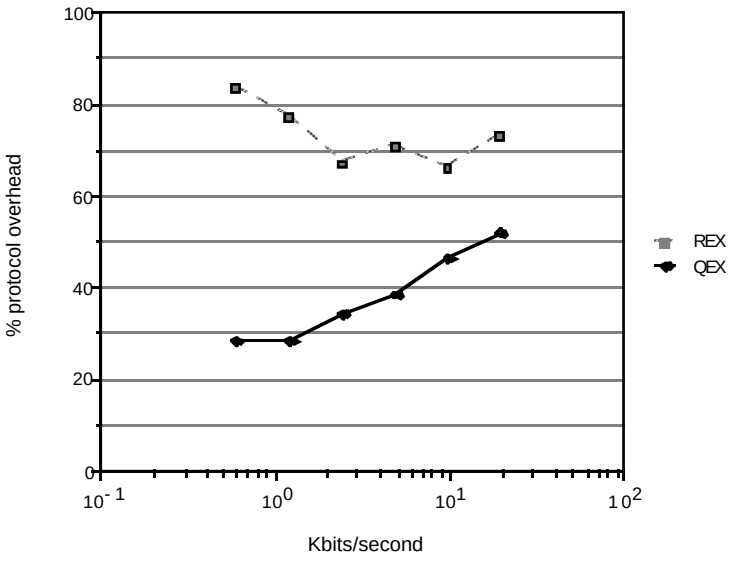

Figure 7.9 - Comparison of the protocol overhead in error channels

This figure highlights the percentage of data lost in protocol overhead against communications bandwidth. The graph clearly shows that QEX maintains a lower protocol overhead for the given payload size over the full range of data rates. The presented figures are calculated with respect to the optimal REX behaviour with no unnecessary retransmissions (that is, sufficient retransmissions to combat packet loss

but no more). At rates below 2.4Kbps, QEX achieves its minima (the least number of packets it could possibly achieve the transmission of the data within). Thus, the overhead of calculating RTTs within QEX can never be less than 28.6% of this payload size. Furthermore, QEX achieves the transmission in the worst case (at 19.2Kbps) with 23.4% overhead above optimal, against 72.9% for REX.

7.5.2.3 Test Case 3 : Fragmented Interactions

The third test case is designed to compare the relative performance of the two protocols for fragmented payloads. The test pair is configured with the same settings as used for the fragmented invocation adaptation testing in section 7.5.1: an invocation with a payload of 2048 bytes is sent (fragmenting into two parts of 1378 and 1026 bytes inclusive of platform headers). A test run consists of 10 invocations. Figure 7.10 illustrates the average time taken for an invocation to complete (averaged over three test runs).

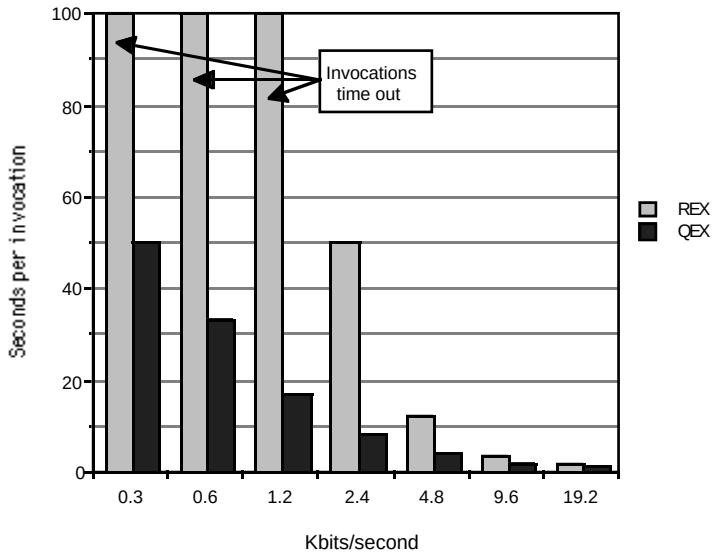

Figure 7.10 - Comparison of time taken to complete one fragmented invocation

The data graphed in the figure shows that QEX consistently outperforms REX for fragmented invocations across the entire range of emulated bandwidths. The REX protocol does not operate any form of congestion control for fragmented invocations. This is catastrophic in the face of reduced bandwidth or network congestion. A single invocation at 2.4Kbps caused over 590 fragments to be queued for transmission. A second invocation would time-out because of the resulting congestion. At lower data rates, the first invocation times out before a single request/reply can get through, leaving over 6,000 fragments queued for transmission. In contrast, once QEX has adapted to the channel, it maintains a consistent rate of two fragments (i.e. the minimum possible) over all the data rates tested.

7.5.2.4 Protocol Analysis

The test cases presented in the preceding sections emphasise different behaviours within QEX. The first test case represents the worst case behaviour of QEX, i.e. REX is performing optimally and the number of messages being used is less than that of QEX. In the more general case, where clients invoke services which take time to complete or the bandwidth is reduced, the optimal behaviour of REX will not be evident. In this case the REX protocol will send a minimum of six messages: CALL, a retry of the CALL, on receipt of the retry the server end will send a CALLACK, a REPLY once the service is complete, a retry of the REPLY if the server has not received a further CALL from the client and a REPLYACK from the client on receipt of the retry of the REPLY. QEX, on the other hand, will still use the same four messages outlined earlier. While the server is processing, both protocols will periodically send a probe message to ensure the operation is continuing. The probe interaction consists of a message header with no data payload which is acknowledged with an appropriate ACK upon reception. Probe messages are sent significantly less often than retries of CALLs or REPLYs and have little impact on the presented figures.

In the more general case, QEX sends less messages than REX, maintaining a higher percentage of user data transferred and using the available network resources more efficiently. However, the additional REX overhead is not reflected in the measured performance of the protocols. This anomaly is primarily due to the result taking mechanism and the multi-threaded nature of ANSAware capsules.

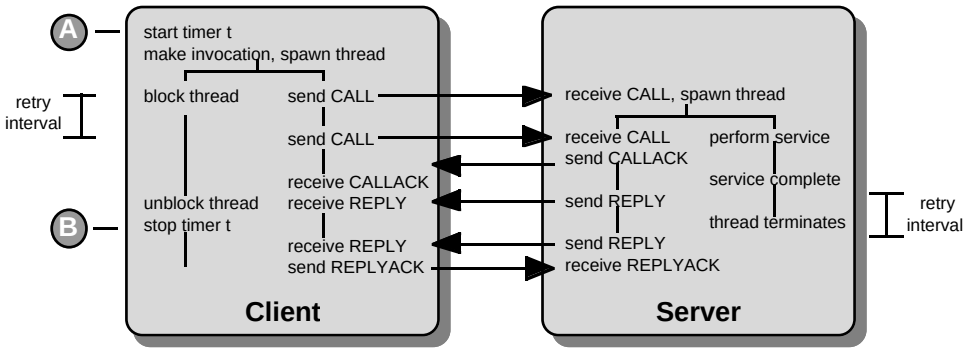

Figure 7.11 - Protocol measurements in detail

Figure 7.11 shows the measurement process for a client and server pair of objects. The client starts the timer to measure the invocation at point A. The client thread becomes blocked and hands over control to the main protocol thread which performs the client part of the invocation. Once the reply is received from the server, the client thread is no longer blocked and therefore able to continue running client code (in this case stopping the timer at point B). As far as the client is concerned, the invocation is

complete. However, the protocol thread will continue after the results are delivered to the client thread, receiving and acknowledging any duplicate REPLYs. This additional messaging continues while the client is effectively busy elsewhere and so does not consciously affect the client or the client side measurements. The additional messages do of course use additional bandwidth and would therefore be expected to be taking resources which could be used by other parties or additional protocol interactions.

On balance, the QEX protocol is faster and more efficient at low bandwidths than REX. Furthermore, QEX is able to adjust to changes in available bandwidth to minimise the number of messages necessary for any given interaction. It would be possible to make QEX compete with REX at higher bandwidths: the variance calculations which are used to detect changes in bandwidth and determine the appropriate backoff strategy could also be used to trigger a “REX mode” which temporarily stopped calculating RTTs to save messages. However, there would clearly be a tradeoff between the minimisation of messages and the speed with which changes in network QoS were determined (see section 6.5.3.1).

The second test case illustrates that further tuning work is required to make QEX more robust to channel errors. At low bandwidths the protocol is seen to use an optimal number of messages which implies the protocol timers are being tuned too conservatively. In contrast, at higher rates the protocol is over compensating for the packet loss and retransmitting unnecessarily (albeit to a far lesser extent than REX).

Lastly, the third test case demonstrates that once QEX has attained a stable state the protocol thoroughly outperforms REX. Moreover, the absence of congestion control within the REX fragmentation mechanism renders it highly unsuitable for bulk data transfers over anything other than lightly loaded high bandwidth networks (i.e. wired WANs or any wireless network).

7.6 Summary

In this chapter the prototype application has been evaluated in terms of feedback gathered subsequent to the limited field trial scenario conducted at the end of the MOST project. The prototype was judged successful in meeting a significant number of the requirements set out in chapter 4. More importantly, the prototype achieved its overall design goal, that is, to provoke interest in developing the concepts further into a product quality system and to further provide a framework for describing the requirements for this new system which will doubtless have a profound effect on the working practices of future field workers.

From a research perspective, the prototype has demonstrated that the concepts of application level adaptation are essential for operation in a heterogeneous networked environment and contributed significantly to the usability of the prototype within the limited context of the test environment. In addition, the computational and engineering perspectives of the supporting platform were evaluated. A number of significant issues have been raised, some of which form the basis of future work discussed in the next chapter.

Finally, the performance of the QEX RPC protocol has been evaluated with respect to the REX protocol. In low bandwidth networks, QEX is found to outperform REX (as expected). Under certain conditions in a fast network, QEX is found to be more expensive than REX. However, the benefits provided by the QoS bindings in allowing the application to adapt are found to outweigh this extra overhead. Chapter 8 presents a series of conclusions and recommends areas of research that require further work.

Chapter 8

Conclusions

8.1 Introduction

This thesis has introduced an infrastructure for supporting advanced applications within the context of a heterogeneous networked environment. More specifically, a new distributed systems platform based on ANSAware has been described which provides a QoS architecture at the computational level and engineering support for this architecture in the form of the QEX protocol. The QoS architecture enables applications to create explicit bindings between platform objects which embody their requirements in terms of a set of new QoS parameters. Moreover, the explicit bindings provide a mechanism through which applications can control and monitor the QoS of platform interactions, facilitating the process of application adaptation. It is argued that, through the process of adaptation, advanced applications are able to operate over a range of networking technologies and still maintain usability. In more detail, the arguments in this thesis are as follows:

Chapter 1 introduced the reasoning behind the development of distributed systems platforms and summarised the most common forms of distribution transparency that are typically supplied by such platforms. A brief introduction was then given to the three main efforts toward the standardisation of open distributed processing systems. The remainder of the chapter discussed the impact of operation within a mobile environment on a typical distributed systems platform and briefly outlined some of the shortcomings with current approaches.

Chapter 2 surveyed the end-system hardware and software architectures and communication technologies that have made the emergence of mobile computing

possible. Based on this survey, the chapter noted the continued improvement in end-system performance and highlighted the emergent trend toward closer integration of end-system and communication technology. Furthermore, given the range of communication technologies now available, it was argued that networks will increasingly be united to provide seamless coverage with dynamic handover between network services (heterogeneous networking).

Chapter 3 provided an in-depth analysis of current research in the field of mobile computing. The chapter highlighted that one of the key characteristics of mobile environments is change. It was noted that the majority of current work seeks to provide abstraction over this change to facilitate mobile working transparently. However, the chapter indicated a number of independent research efforts which have identified that transparency precludes efficient working in all but the simplest applications. The need for facilities to support mobility-aware applications, particularly through adaptation, was highlighted. The chapter concluded by identifying the lack of work on supporting open distributed systems in a mobile context.

Chapter 4 presented background information on the MOST project which provides essential context for discussing the research in this thesis. More specifically, the chapter discusses the organisation, working practices and constraints placed upon a typical utility company. A hypothetical scenario was introduced which was then used to highlight a number of shortcomings that were identified in the working practices of a particular utility company. Subsequently, a short-list was presented of the primary requirements for a prototype system which would help address these shortcomings.

An advanced mobile application was developed that has been designed to satisfy a number of the requirements set out in chapter 4. The application provides a set of tools that enables field workers to collaborate using a variety of forms of information including spatially referenced data such as geographic maps, schematics and technical drawings. The application prototype was discussed at length in chapter 5, with particular emphasis on its group architecture and modular toolbox structure. The chapter also highlighted the most significant design and operational features of the application (including its user interface). The prototype is intended to be capable of operating in a heterogeneous networked environment and the need for support from a suitable distributed systems platform was highlighted.

Chapter 6 studied the new platform infrastructure that was designed to support advanced mobile applications within heterogeneous networked environments. The new platform is based on the ANSAware distributed systems platform. A number of shortcomings with the basic platform were identified. A set of QoS based extensions

were discussed which extend the platform and bring it into closer alignment with the RM-ODP (the RM-ODP and ANSA models are used throughout the chapter as a reference for discussing the new platform). The platform incorporates extensions at the computational and engineering levels to support QoS based explicit bindings. The chapter discussed the binding architecture, QoS parameters and underlying management infrastructure in detail. In addition, a new set of QoS parameters were identified that have been designed specifically for describing platform interactions in these environments. A detailed explanation was given of the QEX protocol which maintains these parameters at the engineering level.

The prototype application and underlying distributed systems platform were evaluated in chapter 7. The evaluation of the application concentrated on how effectively it was perceived to meet the requirements discussed in chapter 4. A limited end-user trial was conducted as part of the MOST project and the feedback from the partner organisation was used to highlight specific limitations within the current prototype. The platform was evaluated qualitatively in terms of the suitability of the paradigm and effectiveness of the execution. Importantly, the chapter also focused on the role of adaptation within the platform and in enabling a number of the prototype's component modules to adapt to changes in the underlying environment. Finally, an analysis of the QEX protocol was presented and the overhead of gathering and managing the QoS information identified through a comparison with the original ANSAware RPC protocol, REX.

This final chapter examines the work presented throughout this thesis and highlights the main results. A number of issues are identified as suitable subjects for future research. Lastly, some concluding remarks are presented.

8.2 Major Results

The major results of the research presented in this thesis are identified in this section. The order in which the results are listed does not imply any relative importance.

8.2.1 Study of Adaptivity

This thesis has highlighted the availability of increasingly diverse and potentially integrated communications infrastructures, and noted the marked expansion in research which aims to exploit this potential. The importance of adaptation is stressed as a key technique for enabling complex applications to operate effectively in changing environments such as these (emphasised by the demands of advanced applications over heterogeneous networked infrastructures). Furthermore, adaptation

is found to be essential at all levels of a system, including the application, middleware, transport and network protocols and, significantly, the user level (e.g. change of work pattern). A number of adaptive techniques which may be applied throughout a system have been identified, including QoS renegotiation, adaptive cache management policies and filtering.

As a basis for investigating adaptation as an approach, an adaptive application and supporting platform which facilitates adaptation has been developed. The platform features a QoS based framework, aligned to the RM-ODP, which allows applications to monitor and control their environment. More importantly, the framework provides an appropriate abstraction for expressing the requirements of the application in terms of QoS. The platform informs the application when changes have occurred in the form of QoS violation up-calls if application specified bounds are exceeded, permitting adaptation. The benefits of the adaptive architecture, with respect to existing non-adaptive approaches, have been clearly demonstrated.

8.2.2 An Explicit QoS Binding Model

The mobile platform infrastructure described throughout this thesis incorporates a number of QoS based extensions which enable it to support adaptive applications. These extensions, modelled as the ability to partially break distribution transparencies, provide mobility support for conventional and mobility-aware applications through the use of implicit and explicit bindings respectively. The thesis uses the platform as a medium for investigating the suitability of the explicit binding mechanism as an architectural tool for creating adaptive applications.

The explicit binding extensions enable computational objects to bind pairs of objects and obtain a handle on to the binding. The handle allows applications to set, query and register for change events in the QoS of the channel specified by the binding. The platform infrastructure is responsible for mapping the binding on to an RPC dialogue and continuously monitoring the activity on the binding. The explicit binding extensions bring the platform into closer alignment with RM-ODP. However, there are a number of significant differences from the RM-ODP model (see section 6.4.2 for a detailed explanation). More specifically :-

- A binding is transient over communication failure, requiring that binding state is managed in locally instantiated monitoring objects. Thus, the two ends of the binding can register independent QoS requirements and present different perspectives on the binding.

- The new model does not allow third parties to establish bindings between objects. However, the binding handle is a reference to a first class interface that may be used by a third party.
- The platform does not support operational causality specifications within interfaces, requiring that a binding is established between a pair of identifying server interfaces (which implicitly offer consumer causality).

The new model was found to be an effective and largely natural way for developing applications which adapt to cope with the problems introduced by mobile environments.

8.2.3 Quality-of-Service Execution Protocol (QEX)

The platform presented in this thesis features a new QoS based RPC protocol called QEX. QEX has been designed to adapt to changes within the supporting network to enable transparent working in a heterogeneous networked environment. More importantly however, QEX incorporates a mechanism for gathering network QoS information which is essential to underpin the remainder of the QoS architecture. Essentially, QEX performs three independent tasks :-

i) Monitoring of packet round-trip times to calculate the channel characteristics.

ii) Adaptation of the protocol retry and backoff strategies in response to changes in network resources.

iii) Feedback to the binder, enabling the policing of application QoS requirements and the generation of violations if QoS bounds are transgressed.

It is the author's intention that the concepts demonstrated by QEX are generic and of general value to the distributed systems community as a whole. It would be of interest to attempt porting elements of the architecture to other distributed systems platforms. For instance, QoS support could be added to implementations of CORBA or DCE. This support would require that equivalent QoS based mechanisms were introduced into the platforms' RPC protocols. Note however that ORBIX [Newman,93], an existing CORBA implementation, uses an RPC mechanism based on TCP and would therefore require QoS monitoring functionality to be added at the transport level.

8.3 Other Significant Results

8.3.1 Investigation of Mobile Collaborative Application Design

This thesis has proposed a set of guidelines for future developers of collaborative mobile applications. In more detail, these guidelines are as follows :-

- Decentralised design for improved availability, fault tolerance and portability across network architectures.
- Modular toolkit design based on a well defined object model to enhance extensibility and ease collaborative development.
- Support for collaboration through application level invocations (as opposed to a generic windowing system approach such as X) for improved interoperability and reduced communications bandwidth.
- Mobility-aware adaptation enabling the same application to run over a dynamic network infrastructure.

These guidelines are the result of developing an advanced application to support field workers within the utilities industries. The application has been used as a framework for evaluating the QoS architecture. In addition, the prototype has demonstrated, through the novel use of colour to reflect changes in the underlying network, that by providing feedback to the user, who is after-all the person responsible for generating the majority of the network traffic, users can implicitly adapt to minimise demands placed on the network.

8.3.2 Establishment of Novel QoS Parameters

This thesis has identified that established QoS parameters, which have been developed for fixed environments, are not sufficient for describing platform invocations in a mobile context. As a starting point for finding a solution, this thesis proposes a basic set of QoS parameters designed to address the problems commonly faced during operation in a mobile environment. This set includes the available throughput and propagation delay, together with novel parameters such as idle time and reachability.

Idle time The idle time enables an object to detect when an invocation has not arrived at an interface within a specified threshold.

Reachability Reachability, strengthens the idle time semantics, by asserting that the absence of contact within the time threshold was not due to unavailability or failure of communications.

Additional work is needed to identify whether further parameters are required for discussing the requirements of continuous media in a mobile environment.

8.3.3 Enhanced Communications Protocol API

This thesis argues that the communications APIs offered by existing protocols and the underlying drivers are too restrictive to enable applications to exploit features of the supporting communications technologies. For instance, preventing applications from optimising for packet-oriented versus connection-oriented services. An extended API is presented which incorporates some additional features :-

QoS interface

Applications are able to interrogate the device to discover the style of the service being provided and a set of QoS parameters describing the tariff structure (if applicable).

Dialling policy support

Applications may register policies to control the dialling and hang-up of connection-oriented services. This enables applications to optimise the link depending on the charging structure in force and the application's requirements.

Deadline driven

The API offers a simple method based on deadlines for multiplexing the different types of information that are typically used by advanced applications.

The new API has been implemented as part of the platform support architecture. It is hoped that as the demands of adaptive advanced applications are recognised, future protocols will offer new facilities to permit adaptation and the efficient multiplexing of multimedia data.

8.4 Future Work

While building the platform infrastructure described in this thesis several issues have come to light which merit further investigation. A number of the more significant points are outlined below.

8.4.1 Integrating Continuous Media

One of the original requirements identified in chapter 4 was to provide audio conferencing facilities between engineers in the field. Due to the physical size of the

then current low-rate vocoder technology, the constraints imposed by the size of the end-systems and the budget of the project, this requirement was not met by MOST. Integrating continuous media such as audio into the platform would require some important extensions :-

Stream bindings and QoS

The current platform provides support for invocation of *operation* interfaces (in RM-ODP terminology). Some work has been presented which describes RM-ODP compatible services for the transmission of continuous media. However, more work is clearly required to integrate continuous media into the distributed systems platform and identify a set of QoS parameters for inclusion in the overall QoS framework. In terms of RM-ODP, this would require the provision of *stream* interfaces and QoS signatures to describe the streams called *flow specifications*.

Efficient transport and multiplexing

The transport of continuous media information places a considerable number of constraints on the protocols and transport services. In particular, packets must be delivered in a timely fashion and in the correct sequence in order to be played intelligibly without noticeable jitter. Such constraints require intelligent scheduling algorithms, both within the platform itself and in the protocol stack and operating system. In addition, the media must be delivered without catastrophically affecting the performance of the rest of the system. Typical protocols will be required to multiplex between different types of traffic with individual requirements, such as streams of continuous media, interactive application traffic and background transfer traffic.

In addition, the integration of continuous media will benefit from a consideration of results from the multimedia community. For example, work on real time scheduling [Steinmetz,95].

8.4.2 New QoS Parameters

This thesis has identified a set of QoS parameters which are used to facilitate adaptation. However, these parameters are not complete. For example, the author envisions that cost and power will place an ever increasing role in future systems. These issues are considered in more detail below.

Cost

Portable machines will be required to interact with an ever wider range of wireless network technologies, particularly as applications become more advanced. It seems most likely that future wide-area wireless technologies, like current cellular systems, will charge for their services. However, with the increased commercial interest in the Internet, it is feasible that wide-area networking in general will become a commercial enterprise. More than ever before, applications will be required to be competitive not just in terms of initial purchase price but also in terms of running cost. In order to become cost effective, applications will be driven to make cost efficient decisions of how to use network resources. To achieve these cost optimisations, information about the services in use must be made available to applications. S-UDP incorporates rudimentary support for describing the cost associated with dial-up channels. However, there is a need to identify a more comprehensive set of QoS parameters which describe the wide range of possible charging strategies used by different services. Furthermore, as networks become more integrated, the charges for each stage of the journey must be accumulated and the QoS parameters will have to reflect how the application may best optimise for the range of technologies.

Power

Current portable machines offer power saving architectures (discussed briefly in chapter 2). These architectures typically provide information on the status of the internal batteries in order to warn when charging is needed. Power saving architectures and improved battery technologies will to some extent reduce the battery consumption problems experienced by high-end portables. However, by providing applications and system software with information about the power consumption of system components, considerable savings could be made. For example, an application might cache information to avoid spinning up the disk, batch information to reduce the power drain of radio transmitters and, potentially, tradeoff different network technologies with different rates of power consumption.

There is considerable scope for investigating how best to represent cost and power, particularly within heterogeneous networked environments, and how applications should adapt when provided with this information.

8.4.3 Improving the RPC Mechanism

The QEX RPC protocol described in chapter 6 is essential for operation of the platform in a mobile environment. The protocol is responsible for gathering round-trip time data and extrapolating the QoS of the channel. The accuracy of the current algorithms is sufficient for the needs of the application prototype. However, the estimations are far from perfect. Further work is required to develop more sophisticated algorithms which are able to filter out the jitter introduced by running over a non-real time operating system and protocol stack.

In addition, the current algorithms assume a symmetric communications channel (equal capacity in both directions) which is not necessarily the case. Modern TDMA systems allow flexible allocation of time slots in each direction. Furthermore, cable television networks offer highly asymmetric channels with extremely high bandwidth down-links and comparatively meagre up-links. New algorithms would be required to cope with asymmetric channels such as these.

RPC and transport protocols commonly deal with network congestion by using exponential backoff strategies in conjunction with rate control mechanisms. However, exponential strategies lead to poor performance in wireless networks as the packets dropped as a result of the increased level of bit error rates are interpreted as congestion. As previously discussed, the strategy within QEX selects between an exponential and a linear strategy depending on the characteristics of the network. Future work is required to find a suitable backoff mechanism to deal with multi-hop networks which have both wired and wireless components.

8.4.4 Investigation of New Communications Primitives

A number of problems have been identified with using synchronous communications primitives, such as RPC, in mobile environments. The problems are concerned primarily with performance, bandwidth efficiency and failure transparency. A number of solutions have been proposed to cope with the frequent connection failure that is often experienced. For example, one approach logs RPCs to persistent storage and replays them upon reconnection (M-RPC advocates this method). Other researchers have postulated that synchronous RPC mechanisms are not well suited to operation in a mobile environment and have proposed more asynchronous styles of interaction. The queued RPC mechanism of the Rover toolkit (essentially asynchronous messaging) is one such approach. The author believes that there is still a need to find a paradigm which offers the required benefits while retaining semantic simplicity. Recent work at Lancaster is beginning to examine a novel communications paradigm based on a Tuple Space model [Gelernter,85].

8.4.5 Group RPC Mechanisms

Current group RPC mechanisms provide transparent point-to-multipoint invocation mechanisms to applications. These mechanisms often rely on highly synchronous token passing algorithms that are unsuitable for most mobile environments. In an integrated heterogeneous networked environment, members of the group may be available via a range of technologies, with consequently, a range of independent qualities-of-service. In many cases only a subset of the group will be available at any one time.

New mechanisms are required which adapt to make the most effective use of the technologies in operation, for instance, exploiting lower level multicast mechanisms. In addition, there is a requirement for a selectively transparent mechanism that provides information back to applications allowing them to adapt. For instance, an application might tradeoff consistency to increase the performance of the group interaction. These mechanisms will be required to provide a suitable set of QoS parameters for describing application requirements such as quorum, timeliness, consistency, ordering and cost. In addition, within such a weakened consistency model, new synchronisation models will also be required. Work is currently underway in developing a selectively transparent group management service [Cheverst,96].

8.5 Concluding Remarks

Future environments will offer a seamless integration of heterogeneous networking technologies providing connectivity regardless of physical location. Whether computers maintain their status as personal computation devices or fulfil Mark Weiser's vision of ubiquitous computing [Weiser,93], it seems clear that the applications running on these machines will experience a rich network infrastructure offering a wide range of services and therefore qualities of service.

This thesis has sought to demonstrate that only through the process of adaptation can applications make the best use of their supporting environment. The platform described herein provides the necessary support for building adaptive applications. Furthermore, it is hoped that the concepts advocated throughout this thesis will, in the long term, lead to the development of adaptive platforms for future environments.

References

[Abdel-Wahab,91] Abdel-Wahab, H.M., and M.A. Feit. “XTV: A Framework for Sharing X Window Clients in Remote Synchronous Collaboration.” *Proc. IEEE Tricomm '91: Communications for Distributed Applications and Systems,* Chapel Hill, April 1991.

[Adams,93] Adams, N., R. Gold, B. Schilit, M. Tso, and R. Want. “The ParcTab Mobile Computing System.” *Proc. 4th Workshop on Workstation Operating Systems (WWOS-IV),* Napa, California, U.S., 14-15 October 1993.

[Adcock,94] Adcock, P., G.S. Blair, and D. Hutchison. “ANSAware and DCE - A Comparison”, *Technical Report MPG-96-23*, Department of Computing, Lancaster University, Bailrigg, Lancaster, U.K. May 1994.

[Amir,95a] Amir, E., H. Balakrishnan, S. Seshan, and R. Katz. “Efficient TCP over Networks with Wireless Links.” *Proc. 5th IEEE Workshop on Hot Topics in Operating Systems (HotOS-V),* Rosario Resort, Orcas Island, Washington, U.S., IEEE Computer Society Press, 4-5 May 1995.

[Amir,95b] Amir, E., S. McCanne, and H. Zhang. “An Application Level Video Gateway.” *Proc. ACM Multimedia,* Hyatt Regency (Embarcadero), San Fransisco, California, U.S., ACM Press, 5-9 November 1995.

[Anderson,90] Anderson, D.P., S. Tzou, R. Wahbe, R. Govindan, and M. Andrews. “Support for Continuous Media in the Dash System.” *Proc. 10th International Conference on Distributed Computing Systems (ICDCS),* Paris, France, May 1990.

[APM,89] Architecture Projects Management Ltd. “The ANSA Reference Manual Release 01.00”, Architecture Projects Management Ltd., Cambridge, U.K. March 1989.

[APM,92] Architecture Projects Management Ltd. “An Introduction to ANSAware 4.0”, Architecture Projects Management Ltd., Cambridge, U.K. February 1992.

[APM,93] Architecture Projects Management Ltd. “Application Programming in ANSAware 4.1”, *Reference Manual RM.102.02*, Architecture Projects Management Ltd., Cambridge, U.K. February 1993.

[Baker,96] Baker, M., X. Zhao, S. Cheshire, and J. Stone. “Supporting Mobility in MosquitoNet.” *Proc. USENIX Annual Technical Conference,* San Diego, California, U.S., 22-26 January 1996.

[Bakre,95a] Bakre, A., and B.R. Badrinath. “I-TCP: Indirect TCP for Mobile Hosts.” *Proc. 15th International Conference on Distributed Computing Systems (ICDCS),* Vancouver, British Columbia, Pages 136-143. 30 May - 2 June 1995.

[Bakre,95b] Bakre, A., and B.R. Badrinath. “M-RPC: A Remote Procedure Call Service for Mobile Clients”, *Technical Report WINLAB TR-98*, Department of Computer Science, Rutgers University, U.S. June 1995.

[Bey,93] Bey, C., B. Anders, J. Perry, and J. Simonoff. “Newton Programmer's Guide.” Apple Computer Inc. Cupertino, CA. 1993.

[Birman,89] Birman, K., and K. Marzullo. “ISIS and the META Project.” *Sun Technology*, Vol. 2 No. 3. September 1989.

[Birman,90] Birman, K.P., and R. Cooper. “The ISIS Project: Real experience with a fault tolerant programming system.” *Proc. ACM/SIGOPS European Workshop on Fault Tolerance Techniques in Operating Systems,* Bologna, Italy, ACM Press, 3-5 September 1990.

[Birrell,84] Birrell, A., and B. Nelson. “Implementing Remote Procedure Calls.” *ACM Transactions on Computer Systems*, Vol. 2 No. 1, Pages 39-59. February 1984.

[Borenstein,93] Borenstein, N., and N. Freed. “MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies”, *RFC 1521*, Internet Engineering Task Force Network Working Group. September 1993.

[Brown,95] Brown, K., and S. Singh. “RelM: Reliable Multicast for Mobile Networks”, *Technical Report*, Computer Science Department, University of South Carolina, Columbia, SC 29208. September 1995.

[Cáceres,94] Cáceres, R., and L. Iftode. “The Effects Of Mobility on Reliable Transport Protocols.” *Proc. 14th International Conference on Distributed Computer Systems (ICDCS),* Poznan, Poland, Pages 12-20. 22-24 June 1994.

[Campbell,93] Campbell, A., G. Coulson, and D. Hutchison. “A Multimedia Enhanced Transport Service in a Quality of Service Architecture.” *Proc. 4th International Workshop On Network and Operating System Support for Digital Audio and Video (NOSSDAV),* Lancaster House, Lancaster, U.K., Editor: D. Shepherd, G. Blair, G. Coulson, N. Davies and F. Garcia, Springer-Verlag, Vol. 846, Pages 124-137. November 1993.

[Cheverst,96] Cheverst, K., N. Davies, A. Friday, and G.S. Blair. “Services to Support Consistency in Mobile Collaborative Applications.” *Proc. 3rd International Workshop on Services in Distributed Networked Environments (SDNE),* Macau, China, IEEE Computer Society Press, Pages 27-34. 3-4 June 1996.

[Cho,94] Cho, K., and K. Birman. “A Group Communication Approach for Mobile Computing.” *Proc. Workshop on Mobile Computing Systems and Applications (MCSA),* Santa Cruz, California, U.S., December 1994. Editor: Luis-Felipe Cabrera and Mahadev Satyanarayanan, IEEE Computer Society Press, Pages 95-102.

[Comer,95] Comer, D., J. Lin, and V. Russo. “An Architecture For A Campus-Scale Wireless Mobile Internet”, *Technical Report CSD-TR 95-058*, Purdue University, Computer Science Building, West Lafayette, IN 47903-1398. 1995.

[Cormen,90] Cormen, T.H., C.E. Leiserson, and R.L. Rivest. “Introduction to Algorithms.” MIT Press. ISBN 0-262-03141-8. 1990.

[Coulson,92] Coulson, G., G.S. Blair, N. Davies, and N. Williams. “Extensions to ANSA for Multimedia Computing.” *Computer Networks and ISDN Systems*, Vol. 25 Pages 305-323. 1992.

[Cross,93] Cross, A.D., J.R. Brailsford, and A.T. Brint. “Expert Systems to Support Network Switching.” *Proc. 12th International Conference on Electricity Distribution (CIRED),* Birmingham, U.K., 17-21 May 1993.

[Danthine,92] Danthine, A., Y. Baguette, G. Leduc, and L. Léonard. “The OSI 95 Connection-mode Transport Service - The Enhanced QoS.” *Proc. 4th IFIP Conference on High Performance Networking,* Liege, Belguim, Elsevier Science Publishers BV (North Holland), Pages 232-252. 14-18 December 1992.

[Davies,94] Davies, N., S. Pink, and G.S. Blair. "Services to Support Distributed Applications in a Mobile Environment." *Proc. 1st International Workshop on Services in Distributed and Networked Environments (SDNE),* Prague, Czech Republic, June 1994.

[Davies,95a] Davies, N., G.S. Blair, K. Cheverst, and A. Friday. "A Network Emulator To Support the Development of Adaptive Applications." *Proc. 2nd USENIX Symposium on Mobile and Location-Independent Computing (MLIC),* Ann Arbor, Michigan, U.S., 10-11 April 1995.

[Davies,95b] Davies, N., G.S. Blair, K. Cheverst, and A. Friday. "Supporting Collaborative Applications in a Heterogeneous Mobile Environment." *Special Issue of Computer Communications on Mobile Computing*, Vol. 19 Pages 346-358. 1995.

[Deering,95] Deering, S., and R. Hinden. "Internet Protocol, Version 6 (IPv6) Specification", *IETF Draft Standard draft-ietf-ipngwg-ipv6-spec-01,* Internet Protocol Next Generation Working Group. March 1995.

[Delgrossi,93] Delgrossi, L., R.G. Herrtwich, C. Vogt, and L.C. Wolf. "Reservation Protocols for Internetworks: A Comparison of ST-II and RSVP." *Proc. 4th International Workshop On Network and Operating System Support for Digital Audio and Video (NOSSDAV),* Lancaster House, Lancaster, U.K., Editor: D. Shephard, G. Blair, G. Coulson, N Davies and F. García, Springer-Verlag, Vol. 846, Pages 199-207. November 1993.

[Demers,94] Demers, A., K. Petersen, M. Spreitzer, D. Terry, M. Theimer, and B. Welch. "The Bayou Architecture: Support for Data Sharing Among Mobile Users." *Proc. Workshop on Mobile Computing Systems and Applications (MCSA),* Santa Cruz, California, U.S., December 1994. Editor: Luis-Felipe Cabrera and Mahadev Satyanarayanan, IEEE Computer Society Press, Pages 2-7.

[Dix,95] Dix, A. "Cooperation without (reliable) Communication." *Proc. IEE Symposium on mobile computing and its applications,* Savoy Place, London, IEE, Vol. 95/219, Pages 4/1-4/4. 24 November 1995.

[Droms,93] Droms, R. "Dynamic Host Configuration Protocol", *RFC 1541*, Internet Engineering Task Force. October 1993.

[Duchamp,91] Duchamp, D., S.K. Feiner, and G.Q. Maguire. "Software Technology for Wireless Mobile Computing", IEE Network Magazine. Vol. 5, Issue 6. Pages 12-18. November 1991.

[ETSI,93] ETSI. “Binary Interchange of Information and Signaling (BIIS) at 1200bit/s”, *I-ETS 300 230*, European Telecommunications Standards Institute. 1993.

[Floyd,95] Floyd, S., V. Jacobson, S. McCanne, C. Liu, and L. Zhang. “A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing.” *Proc. ACM SIGCOMM*, Cambridge, MA, ACM Press, Pages 342-356. August 1995.

[Frazer,87] Frazer, E.L., I. Harris, and P. Munday. “CDLC - A Data Transmission Standard for Cellular Radio.” *Journal of the IERE*, Vol. 57 No. 3, Pages 129-133. 1987.

[Fulton,93] Fulton, J., and C. Kantarjiev. “An Update on Low Bandwidth X (LBX).” *The X Resource*, Vol. 5 No. 1, Pages 251-266. January 1993.

[García,96] García, F., D. Hutchison, A. Mauthe, and N. Yeadon. “QoS Support for Distributed Multimedia Communications.” *Proc. International Conference on Distributed Platforms*, Dresden, 27 February - 1 March 1996.

[Gelernter,85] Gelernter, D. “Generative Communication in Linda.” *ACM Transactions on Programming Langauges and Systems*, Vol. 7 No. 1, Pages 80-112. January 1985.

[Gifford,88] Gifford, D., and N. Glasser. “Remote Pipes and Procedures for Efficient Distributed Communication.” *ACM Transactions on Computer Systems*, Vol. 6 No. 3, Pages 258-283. February 1988.

[GPS,92] GPS. “Federal Radionavigation Plan (FRP)”, *Technical Report DOT-VNTSC-RSPA-92-2/DOD-4650.5*, National Technical Information Service. 1992.

[Greenberg,91] Greenberg, S., and R. Bohnet. “GroupSketch: A multi-user sketchpad for geographically-distributed small groups.” *Proc. Graphics Interface '91*, Calgary, Alberta, Canada, Pages 207-215. 1991.

[Grönvall,96] Grönvall, B., I. Marsh, and S. Pink. “A Multicast-based Distributed File System for the Internet.” *Proc. 7th ACM SIGOPS European Workshop*, Connemera, Ireland, ACM Press, 2-4 September 1996.

[Guy,90] Guy, R., J. Heidemann, W. Mak, T. Page, G. Popek, and D. Rothmeier. “Implementation of the Ficus Replicated File System.” *Proc. USENIX Annual Technical Conference*, Pages 63-71. June 1990.

[Hager,93] Hager, R., A. Klemets, G.Q. Maguire, M.T. Smith, and F. Reichert. "MINT - A Mobile Internet Router." *Proc. 43rd IEEE Vehicular Technologies Conference (VTC),* Secaucus, New Jersey, U.S., IEEE Computer Society Press, May 1993.

[Heron,92] Heron, A., and N. MacDonald. "Proceedings IEE International Conference on Image Processing and its Applications." *Proc. 4th IEE International Conference on Image Processing and its Applications,* Maastricht, The Netherlands, Pages 621-624. April 1992.

[Herrmann,88] Herrmann, F., F. Armand, M. Rozier, M. Gien, V. Abrossimov, I. Boule, M. Guillemont, P. Léonard, S. Langlois, and W. Neuhauser. "CHORUS, a New Technology for Building UNIX Systems." *Proc. UUG Autumn Conference,* Cascais, Portugal, October 1988.

[Hirschmann,96] Hirschmann, M. "A Header Compression Extension to a Remote Procedure Call Protocol for Adaptive Mobile Applications", *Masters Dissertation*, Universitat Karlsruhe and Lancaster University. Telecooperation Office, Institut für Telematik, Vincenz-Prießnitz Straße 1. 1996.

[Honeyman,91] Honeyman, P. "Taking a LITTLE WORK Along", *Technical Report 91-5*, CITI, University of Michigan, Ann Arbor, Michigan, U.S. August 1991.

[Huizinga,94] Huizinga, D.M., and K.A. Heflinger. "Experience with Connected and Disconnected Operation of Portable Notebook Computers in Distributed Systems." *Proc. Workshop on Mobile Computing Systems and Applications (MCSA),* Santa Cruz, California, U.S., December 1994. Editor: Luis-Felipe Cabrera and Mahadev Satyanarayanan, IEEE Computer Society Press, Pages 119-123.

[Huston,95] Huston, L.B., and P. Honeyman. "Partially Connnected Operation." *Proc. 2nd USENIX Symposium on Mobile and Location-Independent Computing (MLIC),* Ann Arbor, Michigan, U.S., Pages 91-97. 10-11 April 1995.

[Imielinski,94a] Imielinski, T., and B.R. Badrinath. "Wireless Mobile Computing: Challenges in Data Management." *Communications of the ACM*, October 1994.

[Imielinski,94b] Imielinski, T., and S. Viswanathan. "Adaptive Wireless Information Systems." *Proc. SIG in Data Base Systems Conference,* Tokyo, Japan, October 1994.

[Ioannidis,93] Ioannidis, J., and G.Q. Maguire. "The Design and Implementation of a Mobile Internetworking Architecture." *Proc. USENIX Winter Conference,* January 1993.

[ISO,92] ISO. “Draft Recommendation X.903: Basic Reference Model of Open Distributed Processing”, *Draft Report*, ISO WG7 Commitee. November 1992.

[ISO,87] ISO. “Information Processing Systems - Data Communications - High Level Data Link Control Element of Procedures”, *ISO 4335*, ISO. 1987.

[ISO,93] ISO. “MPEG-2 Systems Working Draft, Part 2 : Video”, *Draft Report ISO/IEC/JTC1/SC29/WG11 N0601*, MPEG Systems Working Committee. November 1993.

[ISO,95a] ISO. “International Standard ITU-T Recommendation X.903: Open Distributed Processing Reference Model Part 3: Architecture”, *Standard Recommendation ISO/IEC 10746-3: 1995*, ISO WG7 Commitee. January 1995.

[ISO,95b] ISO. “SC21/1 Basic QoS Framework”, *Standard Recommendation ISO IEC JTC 1/SC21*, ISO. 1995.

[Jacobson,90] Jacobson, V. “Compressing TCP/IP Headers for Low-Speed Serial Links”, *RFC 1145*, NIC. February 1990.

[Jacobson,92a] Jacobson, V. “A Portable, Public Domain Network 'Whiteboard'.” *Viewgraphs*, April 1992.

[Jacobson,92b] Jacobson, V., R. Braden, and D. Borman. “TCP Extensions for High Performance”, *RFC 1323*, Internet Engineering Task Force Network Working Group. May 1992.

[Johansen,95] Johansen, D., R. van Renesse, and F. Schneider. “Operating System Support for Mobile Agents.” *Proc. 5th IEEE Workshop on Hot Topics in Operating Systems (HotOS-V)*, Rosario Resort, Orcas Island, Washington, U.S., IEEE Computer Society Press, 4-5 May 1995.

[Joseph,95] Joseph, A., A. deLespinasse, J. Tauber, D. Gifford, and M.F. Kaashoek. “Rover: A Toolkit for Mobile Information Access.” *Proc. 15th ACM Symposium on Operating System Principles (SOSP)*, Copper Mountain Resort, Colorado, U.S., ACM Press, Vol. 29, Pages 156-171. 3-6 December 1995.

[Kantarjiev,93] Kantarjiev, C.K., A. Demers, R. Frederick, R.T. Krivacic, and M. Weiser. “Experiences with X in a Wireless Environment.” *Proc. USENIX Symposium on Mobile and Location Independent Computing (MLIC)*, Cambridge, Massachusetts, U.S., Pages 117-128. 2-3 August 1993.

[Käppner,94] Käppner, T., and L. Wolf. “Media Scaling in Distributed Multimedia Object Services.” *Proc. 2nd International Workshop (IWACA),* Heidelberg, Germany, Editor: R. Steinmetz, Springer-Verlag, Vol. 868, Pages 34-43. September 1994.

[Katz,94] Katz, R.H. “Adaptation and Mobility in Wireless Information Systems.” *IEEE Personal Communications*, Vol. 1 No. 1, Pages 6-17. First Quarter 1994.

[Katz,96a] Katz, R., and E. Brewer. “The Case for Wireless Overlay Networks.” *Proc. SPIE Multimedia and Networking Conference (MMNC),* San Jose, California, U.S., 29-30 January 1996.

[Katz,96b] Katz, R., E. Brewer, E. Amir, H. Balakrishnan, A. Fox, S. Gribble, T. Hodes, D. Jiang, G. Nguyen, V. Padmanabhan, and M. Stemm. “The Bay Area Research Wireless Access Network (BARWAN).” *Proc. IEEE COMPCON Spring '96 - 41st International Computer Conference,* Santa Clara, California, U.S., IEEE Press, 25-28 February 1996.

[Keeton,93] Keeton, K., B.A. Mah, S. Seshan, R.H. Katz, and D. Ferrari. “Providing Connection-Oriented Network Services to Mobile Hosts.” *Proc. USENIX Symposium on Mobile and Location Independent Computing (MLIC),* Cambridge, Massachusetts, U.S., Pages 83-102. 2-3 August 1993.

[Kistler,91] Kistler, J.J., and M. Satyanarayanan. “Disconnected Operation in the Coda File System.” *Proc. 13th ACM Symposium on Operating Systems Principles (SOSP),* Asilomar Conference Center, Pacific Grove, U.S., ACM Press, Vol. 25, Pages 213-225. 13-16 October 1991.

[Kojo,94] Kojo, M., K. Raatikainen, and T. Alanko. “Connecting Mobile Workstations to the Internet over a Digital Cellular Telephone Network”, *Technical Report C-1994-39,* University of Helsinki, Department of Computer Science, P.O. Box 26, FIN-00014. September 1994.

[Kuenning,94] Kuenning, G. “Design of the SEER Predictive Caching Scheme.” *Proc. Workshop on Mobile Computing Systems and Applications (MCSA),* Santa Cruz, California, U.S., December 1994. Editor: Luis-Felipe Cabrera and Mahadev Satyanarayanan, IEEE Computer Society Press, Pages 37-43.

[Lai,95] Lai, S., A. Zaslavsky, G. Martin, and L. Yeo. “Cost Efficient Adaptive Protocol with Buffering for Advanced Mobile Database Applications.” *Proc. International Conference on Database Systems for Advanced Applications,* Singapore, 10-13 April 1995.

[Lantz,86] Lantz, K.A. “An experiment in integrated multimedia conferencing.” *Proc. Computer-Supported Cooperative Work (CSCW)*, Austin, Texas, U.S., December 1986.

[Lauwers,90] Lauwers, J.C., and K.A. Lantz. “Collaboration Awareness in Support of Collaboration Transparency: Requirements for the Next Generation of Shared Window Systems.” *Proc. Computer Human Interaction (CHI)*, Pages 303-310. 1990.

[Li,94] Li, G. “Real-Time ANSAware (RAW) Version 1.0: Programming and System Overview”, *RFC APM.1207.00.03*, Architecture Projects Management Cambridge Limited, UK. July 1994.

[Mason,89] Mason, R., and A. Kaye. “Mindweave: Communication, Computers and Distance Education.” Oxford: Pergamon Press. 1989.

[McCaw,93] McCaw Cellular et al. “CDPD System Specifications”, *Preliminary release 0.8*, McCaw Cellular. March 1993.

[Morgan,91] Morgan, T. “Lightning Flash Location System”, *Product Announcement ERDC 1056/02.91*, Electricity Research and Development Centre, EATL, Capenhurst, Chester, U.K. February 1991.

[MOST,92] MOST. “MOST: Mobile Open Systems Technology for the Utilities Industry”, *Project Proposal* Lancaster University and EA Technology. Department of Computing, Lancaster University, Bailrigg, Lancaster, U.K. 1992.

[Motorola,95] Motorola. “Marco and Envoy Wireless Communicators.””Motorola Inc., Corporate Offices, 1303 E. Algonquin Rd., Schaumberg, IL 60196. Motorola Inc. 1995.

[Motorola,96] Motorola. “Tango Two-way Pager.””Motorola Inc., Corporate Offices, 1303 E. Algonquin Rd., Schaumberg, IL 60196. Motorola Inc. 26 July 1996.

[Mummert,95] Mummert, L., M. Ebling, and M. Satyanarayanan. “Exploiting Weak Connectivity for Mobile File Access.” *Proc. 15th ACM Symposium on Operating System Principles (SOSP)*, Copper Mountain Resort, Colorado, U.S., ACM Press, Vol. 29, Pages 143-155. 3-6 December 1995.

[Myles,93a] Myles, A., and D. Skellern. “Comparing four IP based mobile host protocols.” *Proc. 4th Joint European Networking Conference*, Trondheim, Norway, Pages 191-196. 10-13 May 1993.

[Myles,93b] Myles, A., and D. Skellern. “Comparison of Mobile Host Protocols for IP.” *Internetworking Research and Experience*, Vol. 4 Pages 175-194. May 1993.

[Newman,93] Newman, C. “The ORBIX Architecture”, *White paper* IONA Technologies Inc., 201 Broadway, Floor 3, Cambridge, MA 02139-1955. 1993.

[Nieh,95] Nieh, J., and M. Lam. “Integrated Processor Scheduling for Multimedia.” *Proc. 5th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV),* Durham, New Hampshire, U.S., 19-21 April 1995.

[Noble,95] Noble, B.D., M. Price, and M. Satyanarayanan. “A Programming Interface for Application-Aware Adaptation in Mobile Computing.” *Proc. 2nd USENIX Symposium on Mobile and Location Independent Computing (MLIC),* Ann Arbor, Michigan, U.S., Pages 57-66. 10-11 April 1995.

[OMG,91] The Object Management Group. “The Common Object Request Broker: Architecture and Specification (CORBA)”, *Technical Report 91.12.1*, The Object Management Group. 1991.

[OSF,91] The Open Software Foundation. “Distributed Computing Environment: An Overview”, *Technical Report*, Open Group Research Institute, 11 Cambridge Center, Cambridge, MA 02142. April 1991.

[Peine,96] Peine, H., T. Stolpmann, and J. Nehmer. “Agents for Remote Actions (ARA)”, *World WideWeb page http://www.uni-kl.de/AC-Nehmer/Ara*, University of Kaiserslautern, Department of Computer Science, P.O. Box 3049, D-67653, Kaiserslautern, Germany. February 1996.

[Perkins,92] Perkins, C., and Y. Rekhter. “Shortcut routing for mobile hosts”, *draft RFC*, IBM. July 1992.

[Perkins,94] Perkins, C., A. Myles, and D. Johnson. “The Internet Mobile Host Protocol.” *Proc. iNET'94*, Prague, Czech Republic, 15-17 June 1994.

[Perkins,95a] Perkins, C. “IP Mobility Support”, *Draft Standard draft-ietf-mobileip-protocol-12*, Internet Engineering Task Force. August 1995.

[Perkins,95b] Perkins, C., and D. Johnson. “Mobility Support in IPv6”, *Technical Report draft-perkins-ipv6-mobility-sup-02*, Internet Engineering Task Force. July 1995.

[Pope,95] Pope, S. “Mobility in a Trading Environment”, *Technical Report* Cambridge University. 27 June 1995.

[Proxim,96] Proxim. “RangeLan II, Product #7200.””Proxim Inc., 295 North Bernado Ave., Mountain View, CA 94043. Proxim Inc. July 1996.

[Reed,79] Reed, D., and R. Kanodia. “Synchronisation with eventcounts and sequencers.” *Communications of the ACM*, Vol. 22 No. 2, Pages 115-123. 1979.

[van Renesse,94] van Renesse, R., T.M. Hickey, and K.P. Birman. “Design and Performance of Horus: A Lightweight Group Communications System”, *Technical Report TR 94-1442*, Department of Computer Science, 4130 Upson Hall, Cornell University, Ithaca, NY 14853-7501. August 1994.

[Rodden,92] Rodden, T. “A Survey of CSCW Systems”, Lancaster University. 1992.

[Satyanarayanan,85] Satyanarayanan, M., J.H. Howard, D.N. Nichols, R.N. Sidebotham, A.Z. Spector, and M.J. West. “The ITC Distributed File System: Principles and Design.” *Proc. 10th Symposium on Operating System Principles (SOSP)*, Orcas Island, Washington, U.S., ACM Press, December 1985.

[Satyanarayanan,90] Satyanarayanan, M., J.J. Kistler, P. Kumar, M.E. Okasaki, E.H. Siegel, and D.C. Steere. “Coda: A Highly Available File System for a Distributed Workstation Environment.” *IEEE Transactions on Computers*, Vol. 39 No. 4, Pages 447-459. April 1990.

[Schill,95] Schill, A., B. Bellmann, W. Bohmak, and S. Kummel. “System Support for Mobile Distributed Applications.” *Proc. 2nd International Workshop on Services in Distributed and Networked Environments (SDNE)*, Whistler, British Columbia, IEEE Computer Society Press, Pages 124-131. June 1995.

[Schoute,80] Schoute, P.C. “Control of Aloha Signalling in a Mobile Radio Trunking System.” *Proc. IEE Conference on Radio Spectrum Conservation Techniques*, Savoy Place, London, IEE, July 1980.

[Seal,96] Seal, K., and S. Singh. “Loss Profiles : A Quality of Service Measure in Mobile Computing.” *Wireless Computing*, Vol. 2 Pages 45-61. 1996.

[Seshan,95] Seshan, S. “Low Latency Handoff in Wireless Networks”, *Ph.D thesis*, University of California, Berkeley, California, U.S. 1995.

[Singh,96] Singh, S. “Quality of Service Guarantees in Mobile Computing.” *Computer Communications*, Vol. 19 Pages 359-371. 1996.

[SkyTel,96] SkyTel. “SkyTel 2-Way, Product #MP770101.””SkyTel, 2400 E. Katella Ave., Orange, CA 92667-5233. Skytel. July 1996.

[Socket,96] Socket. “PageCard WMS, Product #PA0501.””Socket/Mitsubishi Corp., 6500 Kaiser Drive, Fremont, CA 94555-3613. Socket/Mitsubishi Corp. July 1996.

[Steinmetz,95] Steinmetz, R., and K. Nahrstedt. “Multimedia: computing, communications, and applications.” Prentice Hall. ISBN 0-13-324435-0. Upper Saddle River, NJ 07458. 1995.

[Sun,88] Sun. “Remote Procedure Call Specification”, *RFC 1050*, Sun Microsystems Inc. June 1988.

[Sun,95] Sun. “The Java Language Specification Version 1.0 Beta”, *Draft Specification*, Sun Microsystems Computer Corporation. October 1995.

[Tait,93] Tait, C. “A File System for Mobile Computing”, *Ph.D thesis*, Columbia University, Department of Computer Science, 1214 Amsterdam Ave., New York, NY 10027-7003. 1993.

[Tait,95] Tait, C., H. Lei, S. Acharya, and H. Chang. “Intelligent File Hoarding for Mobile Computers.” *Proc. 1st ACM International Conference on Mobile Computing (MOBICOM)*, Berkeley, California, U.S., ACM Press, Pages 119-125. 13-15 November 1995.

[Tanenbaum,88] Tanenbaum, A.S. “Computer Networks.” Prentice-Hall. ISBN 0-13-166836-6. 1988.

[Telecom,78] British Telecom. “A standard code for radiopaging”, *Technical Report*, Post Office Code Standardisation Advisory Group, British Telecom Labs., St. Peter’s House, St. Peter’s Street, Colchester, Essex, CO1 1ET. June 1978.

[Teraoka,92] Teraoka, F. “Design Implementation and Evaluation of Virtual Internet Protocol.” *Proc. 12th International Conference on Distributed Computing Systems (ICDCS)*, Pages 170-177. June 1992.

[Teraoka,93] Teraoka, F., and M. Tokoro. “Host Migration Transparency in IP Networks: The VIP Approach”, *Technical Report*, Sony Computer Science Laboratory Inc., Tokyo, Japan. 1993.

[Teraoka,95] Teraoka, F. “Options for Mobility Support in IPv6”, *Technical Report draft-teraoka-ipv6-mobility-sup-01*, Internet Engineering Task Force. June 1995.

[UADG,93] UADG. “Mobile Access Protocol for MPT 1327 Equipment (MAP 27)”, User Access Definition Group. August 1993.

[Vaisey,92] Vaisey, J., E. Yuen, and J. Cavers. “Video Coding for Very High Rate Mobile Data Transmission.” *Proc. 42nd IEEE Vehicular Technologies Conference (VTC)*, IEEE Computer Society Press, Pages 259-262. May 1992.

[Waldspurger,94] Waldspurger, C.A., and W.E. Weihl. “Lottery Scheduling: Flexible Proportional-Share Resource Management.” *Proc. 1st Symposium on Operating Systems Design and Implementation,* Monterey, California, U.S., Pages 1-11. November 1994.

[Watson,94] Watson, T. “Application Design for Wireless Computing.” *Proc. Workshop on Mobile Computing Systems and Applications (MCSA),* Santa Cruz, California, U.S., December 1994. Editor: Luis-Felipe Cabrera and Mahadev Satyanarayanan, IEEE Computer Society Press, Pages 91-94.

[Weiser,93] Weiser, M. “Some Computer Science Issues in Ubiquitous Computing.” *Communications of the ACM*, Vol. 36 No. 7, Pages 74-84. July 1993.

[Westervelt,91] Westervelt, J. “Introduction to GRASS 4”, GRASS Information Center, U.S. Army CERL, Champaign, Illinois, U.S. July 1991.

[White,96] White, J.E. “Mobile Agents.” *in Software Agents.* Editor: J. Bradshaw. Menlo Park, California, U.S.: AAAI Press and MIT Press, Available January 1997.

[Wong,95] Wong, P., and D. Britland. “Mobile Data Communications Systems.” Artech House. ISBN 0-89006-751-1. Norwood, Massachusetts, U.S. 1995.

[Worship,94] Worship, G. “Specification for Application Providing Remote Access to Central Copmputing Facility”, *Internal Report EATL/MOST/TN.16/94-7*, Electricity Research and Development Centre, EATL, Capenhurst, Chester, U.K. July 1994.

[Xircom,96] Xircom. “Netwave Adaptor, Product #CNW.””Xircom Inc., 2300 Corporatate Center Drive, Thousand Oaks, CA 91320. Xircom Inc. July 1996.

[Yeadon,94] Yeadon, N., F. Garcia, A. Campbell, and D. Hutchison. “QoS Adaptation and Flow Filtering in ATM Networks.” *Proc. 2nd International Workshop (IWACA),* Heidelberg, Germany, Editor: R. Steinmetz, Springer-Verlag, Vol. 868, Pages 191-202. September 1994.

[Zenel,95] Zenel, B., and D. Duchamp. “Intelligent Communication Filtering for Limited Bandwidth Environments.” *Proc. 5th IEEE Workshop on Hot Topics in Operating Systems (HotOS-V),* Rosario Resort, Orcas Island, Washington, U.S., IEEE Computer Society Press, 4-5 May 1995.

Appendix A

The Enhanced DPL Language

The DPL language, first introduced in section 6.2.5.1, has been modified to incorporate the new binding and QoS extensions. The DPL language syntax is described by the compiler grammar presented below. The enhancements to the language are highlighted in bold.

```
statement        ::= resultlist handle $ operation argumentlist
                     exceptionlist qos deadline within
                     | { rlist } INITIATES handle $ operation
                       argumentlist qos deadline within
                     | { rlist } INSTANTIATES handle $
                       instantiateop argumentlist
                     | USE TOKEN from
                     | DECLARE declarationlist TOKEN decltype
                     | MEMBERSTATE IS statelist IN TOKEN
                     | STORAGE decltype TOKEN state
                     | SNAPSHOT OF statelist IN TOKEN OF TOKEN
                     | MANAGED alist
                     | GROUPMEMBER TOKEN
                     | resultlist BINDERTOKEN $ handle bindlist
deadline         ::= /* empty */ | DEADLINE TOKEN
bindlist         ::= ( TOKEN , TOKEN , TOKEN )
within           ::= /* empty */ | WITHIN TOKEN
qos              ::= /* empty */ | { TOKEN }
from             ::= /* empty */ | FROM TOKEN
decltype         ::= CLIENT | SERVER
declarationlist  ::= /* empty */ | DECLARATION
                     | { rlist } DECLARATION
state            ::= /* empty */ | STATE statelist
statelist        ::= { VTlist }
```

```
VTlist          ::= /* empty */ | vtlist
vtlist          ::= pair | vtlist , pair
pair            ::= TOKEN TOKEN
resultlist      ::= /* empty */ | BECOMES | { rlist } BECOMES
rlist           ::= /* empty */ | results
results         ::= result | results , result
result          ::= TOKEN
handle          ::= TOKEN
operation       ::= TOKEN | CREATE | RECREATE | DESTROY
                    | CREATEMEMBER | DESTROYMEMBER
instantiateop   ::= CREATE | RECREATE | DESTROY
                    | CREATEMEMBER | DESTROYMEMBER
argumentlist    ::= /* empty */ | ( alist )
alist           ::= /* empty */ | arguments
arguments       ::= argument | arguments , argument
argument        ::= stoken
stoken          ::= TOKEN | QUOTED_STRING
exceptionlist   ::= /* empty */ | exceptionlist continuelist
                    | exceptionlist signallist
                    | exceptionlist abortlist
continuelist    ::= CONTINUE statuslist
statuslist      ::= TOKEN | statuslist , TOKEN
signallist      ::= SIGNAL statuslist
abortlist       ::= ABORT statuslist
```