“A Session Layer Protocol to Manage Multi-homed Mobile Communication”

Stephen Larkin ***BSc. (HONS)***

For the degree of Master of Science by Research

Supervised by Professor Srba Cvetkovic

Departments of Computer Science,
Electronic and Electrical Engineering and
Mechanical Engineering.
University of Sheffield

Declaration

All sentences or passages quoted in this thesis from other people's work have been specifically acknowledged by clear cross-referencing to the author, work and page(s). I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this thesis and the degree examination as a whole.

Name:	Stephen Larkin
Signature:	
Date:	29th August 2000

Acknowledgements

Thank-you to my supervisor, Srba Cvetkovic, who has welcomed me into his research group and given me a good overview of what I needed know. Also thanks go to Matthias Kraner and Chern Nam Yap for their help with the design of the protocol. Special thanks to Emlyn Bolton-Maggs for sharing his knowledge of operating systems and software and for his support during the project. Thanks to James Matthews for proofreading this thesis and for a constant supply of coffee breaks. Finally I would like to thank Kate McDougall and my parents for their love and support.

Abstract

Mobile access to the Internet is set to revolutionise the world in which we live in. However current mobile devices leave a lot to be desired. This work investigates the use of multiple Internet connections from a mobile device (known as multi-homing) to see if uninterrupted communication is possible in the mobile environment where handoff and lack of bandwidth pose continuous problems.

A session layer protocol is designed and implemented and rudimentary testing is carried out to evaluate the performance. The aim of the protocol is to assign each successively initiated communications session to a different network interface and to redirect the flow of communication should that interface hand-off.

The results show that the implemented protocol suffers from an increase in latency by a factor of ten. This is thought to be due to the chosen implementation platform the WinSock Service Provider Interface on Windows 2000.

List of Contents

DECLARATION **2**

ACKNOWLEDGEMENTS **3**

ABSTRACT **4**

LIST OF CONTENTS **5**

LIST OF FIGURES **7**

LIST OF TABLES **7**

LIST OF ABBREVIATIONS **8**

CHAPTER 1 - INTRODUCTION **9**

1.1 PROJECT OUTLINE 9
1.2 AIMS AND OBJECTIVES 10
1.3 SCOPE OF THE WORK 11
1.4 ORGANISATION OF THIS DOCUMENT 11

CHAPTER 2 - BACKGROUND **12**

2.1 CELLULAR NETWORK STRUCTURE 12
2.2 IP NETWORK ARCHITECTURE, ADDRESSING AND ROUTING 14
2.3 MOBILE IP ROUTING 16
2.4 IP BASED CELLULAR NETWORKS 18
2.5 LITERATURE REVIEW 19
2.5.1 Flexible Network Support for Mobility *19*
2.5.2 Wireless Overlay Networks *20*
2.5.3 MSOCKS: An Architecture for Transport Layer Mobility *21*
2.5.4 Handoff in Hybrid Mobile Data Networks *22*
2.5.5 Dynamic Network Reconfiguration Support for Mobile Computers *22*
2.5.6 Session Control Transmission Protocol *23*
2.6 AN INTEGRATED ARCHITECTURE FOR MOBILE QUALITY OF SERVICE, SECURITY AND BILLING 24

CHAPTER 3 - PROTOCOL DESIGN AND IMPLEMENTATION **25**

3.1 REQUIREMENTS OF THE PROTOCOL 25

3.1.1 Interface Selection Requirements 26
3.1.2 Session Redirection Requirements 26
3.2 PROTOCOL DESIGN 28
3.2.1 Placement in the Network Stack 28
3.2.2 Signalling 29
3.2.3 Interface Selection 30
3.2.4 Session Redirection 32
3.2.5 Operation with UDP 35
3.2.6 Operation with TCP 35
3.2.7 Protocol State Diagram 35
3.2.8 Multicast Support 35
3.3 PROTOCOL IMPLEMENTATION 37
3.3.1 Implementation Platform 37
3.3.2 Windows' Sockets Architectural Model and Interfaces 38
3.3.3 The Multi-homed Management Layered Service Provider 41
CHAPTER 4 - PROTOCOL PERFORMANCE 43
4.1 TEST-BED ARCHITECTURE 43
4.2 DESCRIPTION OF TESTS 45
4.3 ANALYSIS OF RESULTS 45
CHAPTER 5 - CONCLUSIONS 48
5.1 SUMMARY OF RESULTS 48
5.2 FUTURE WORK 48
5.3 PROJECT EVALUATION 49
APPENDIX A - REFERENCES 50
APPENDIX B - BIBLIOGRAPHY 53
APPENDIX C - CONTENTS OF ACCOMPANYING DISK 54
APPENDIX D - ORIGINAL PROJECT PROPOSAL 55

List of Figures

FIGURE 1 – FREQUENCY RE-USE IN CELLULAR NETWORKS 13
FIGURE 2 – EXAMPLE CELLULAR NETWORK STRUCTURE (FROM [HAMALAINEN96])........ 14
FIGURE 3 - EXAMPLE IP NETWORK ARCHITECTURE 14
FIGURE 4 – THE IETF MOBILE IP ARCHITECTURE 17
FIGURE 5 - ARCHITECTURE OF AN IP BASED CELLULAR NETWORK 18
FIGURE 6 - MOVEMENT IN AN IP CELLULAR NETWORK... 27
FIGURE 7 – LOCATION OF THE PROTOCOL IN THE NETWORK STACK.................................. 28
FIGURE 8 – SESSION HEADER STRUCTURE... 31
FIGURE 9 – BINDING UPDATE REQUEST HEADER STRUCTURE 32
FIGURE 10 – SESSION LAYER SIGNALLING.. 34
FIGURE 11 – SESSION LAYER STATE DIAGRAM 36
FIGURE 12 – WINSOCK API VS. OSI/RM 38
FIGURE 13 – THE WINSOCK 2 WOSA-COMPLIANT PROGRAMMING INTERFACES (FROM [HUA99]) 39
FIGURE 14 – MULTI-HOMED MANAGEMENT PROTOCOL IN THE WINSOCK NETWORK STACK 40
FIGURE 15 – LAYOUT OF THE TEST-BED 44

List of Tables

TABLE 1 – WINSOCK FUNCTIONS 38
TABLE 2 – MAPPING FROM WINSOCK API FUNCTIONS TO SPI FUNCTIONS 41
TABLE 3 – TEST-BED ADDRESSING 43
TABLE 4 – TEST-BED COMPUTERS 45
TABLE 5 – NETWORK ANALYSER TIMING (NORMAL OPERATION).................................... 46
TABLE 6 – NETWORK ANALYSER TIMING (PROTOCOL INSTALLED).................................. 46
TABLE 7 - WINSOCK MONITOR TIMING (SENDER – NORMAL OPERATION) 46
TABLE 8 - WINSOCK MONITOR TIMING (RECEIVER – NORMAL OPERATION).................... 46
TABLE 9- WINSOCK MONITOR TIMING (SENDER – PROTOCOL INSTALLED) 47
TABLE 10 - WINSOCK MONITOR TIMING (RECEIVER – PROTOCOL INSTALLED)................ 47

List of Abbreviations

API	Application Programming Interface
ARP	Address Resolution Protocol
ATM	Asynchronous Transfer Mode
BSC	Base Station Controller
BTS	Base Transceiver Station
DLL	Dynamic Link Library
DNS	Domain Name System
ETSI	European Telecommunications Standards Institute
FA	Foreign Agent
GPRS	General Packet Radio System
GSM	Global System for Mobile Communications
GTP	Gateway Tunnelling Protocol
HA	Home Agent
IPSec	Internet Protocol Security
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IPX/SPX	Internet Packet Exchange / Sequence Packet Exchange
IrDA	Infrared Data Association
ISO	International Standards Organisation
LSP	Layered Service Provider
MSC	Mobile Switching Centre
NetBIOS	Network Basic Input / Output System
OSI/RM	Open Systems Interconnection Reference Model
PCMCIA	Personal Computer Memory Card International Association
PDU	Protocol Data Unit
PMI	Physical Media Independence
PPP	Point-to-Point Protocol
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RSVP	Resource Reservation Protocol
SCTP	Session Control Transmission Protocol
SID	Session Identifier
SLIP	Serial Line Internet Protocol
SPI	Service Provider Interface
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
WAP	Wireless Application Protocol
WinSock	Windows Sockets
WOSA	Windows Open Service Architecture

Chapter 1 - Introduction

1.1 Project Outline

The area of mobile communications is currently going through a period of immense change comparable to that of the industrial revolution at the turn of the last century. The buzzword is 'convergence' – meaning the marriage of mobile telephony with the Internet. Individually, these two phenomena have enjoyed exponential subscriber growth during the last five years and are set to hit the one billion-subscriber mark by 2004 [UMTSForum99]. However, the real boom will come when global, mobile Internet access becomes established and access to information will be easily available from anywhere on the planet. The current Wireless Application Protocol (WAP) [WAPForum99] is a tiny step in the right direction towards this dream but due to slow data transfer rates and the physical limitations of the devices it is not perfect. The aim of this work is to produce some communications software – commonly known as a protocol – to make the next step in the evolution towards the dream.

In order to make this step it is necessary to look at where technology is now and what direction it is likely to go in the future. The current WAP technology provides only limited access to the Internet by proxy. This means that the user does not have their own Internet Protocol (IP) address and all information is tunnelled to the mobile device inside another network protocol (such as GTP – the Gateway Tunnelling Protocol [ETSI99]). It is widely predicted [Nokia99] that future mobile networks will use the Internet Protocol in the Radio Access Network – i.e. the wireless portion of the network. This will mean that each mobile device will have (at least one) IP address. Data packets will be transferred to the mobile user via existing IP routing mechanisms and newer Mobile IP routing [Perkins96]. This first point emphasises the fact that future mobile networks will be "IP all the way" as opposed to the current method that uses IP to the edge of the mobile network and then tunnels the IP data packets through the mobile network to the user's device. The second point to note about future mobile technology is that wireless networks will always offer limited functionality when compared to their wired counterparts. Bandwidth is always going to be a scarce resource and due to the nature of the cellular architecture there will always be periods where connectivity is lost when moving between cells (known as handoff). In summary the two important points here are:

- Routing in future mobile networks will be based completely on IP;
- Lack of bandwidth and loss of service during handoff will always cause problems.

The aim of this work is to try to alleviate the problems caused by handoff and lack of bandwidth through the use of multiple connections to the Internet from a mobile device. By looking around the countryside the appearance of the cellular base-station has become more noticeable during the last few years. This is by no means a coincidence with the exponential subscriber growth outlined in the first paragraph. Nevertheless this means that one is constantly within the range of a plethora of wireless networks. Currently in the United Kingdom there are four GSM mobile operators with another five Universal Mobile Telecommunications System (UMTS) [UMTSForum99] licenses having just been sold. This means that in the near future there will be nine service providers offering different levels of mobile Internet connectivity. When coupled with the array of private wireless networks such as those operating inside offices or on-campus and the range of short range wireless communications technology – such as Infrared and the forthcoming Bluetooth [Bluetooth99] – this leads us to the fact that we are surrounded by a multiplicity of available wireless connections to the Internet. So why should we use just one?

1.2 Aims and Objectives

It is necessary to have some software – a protocol – operating on the mobile device in order to exploit the full potential of the available wireless networks in our surroundings. Current mobile communications software is built around the assumption that a mobile device has only one network interface and hence do not support multi-homed communication (transmission through more than one network interface). The aim of this work is to design a protocol that will operate above the current transport protocols and manage the allocation of different communications sessions to different network interfaces. Furthermore, if one interface goes through a period of handoff, the session must be redirected through an alternate interface in order to avoid data loss. Finally, once the protocol is developed it is important to analyse the performance of its operation. These rather general aims can be split into a number of objectives.

1. Design the protocol to manage multi-homed communication,
2. Implement the protocol on Microsoft Windows 2000 as a WinSock 2 Service Provider so that legacy applications can operate transparently,

3. Set up an experimental network test bed to simulate the operation of the protocol in a wireless environment,
4. Measure the delay caused by the protocol operation in terms of the delay caused by the added function calls in the session layer,
5. Measure the delay for the redirection of communication sessions between interfaces during a simulated handoff,
6. Draw conclusions about the performance and operation of the protocol.

1.3 Scope of the work

At this point it is important to state the scope of this work. Multi-homed mobile communication is a new idea and this protocol is the first stage of research into how to manage multiple network interfaces. As such, the scope of this project is to design and implement an initial protocol and to assess its performance.

1.4 Organisation of this Document

The structure of the rest of this document is as follows. Chapter two introduces the relevant background information that is necessary for reading the rest of the document. The chapter also includes a literature review that gives an overview of any work similar to this and explains how this work differs. Chapter three describes the protocol design and implementation stage. It includes state diagrams for operation above the Internet UDP and TCP protocols and timing diagrams to show how signalling is carried out. The implementation details describe the software design decisions and the implementation platform specifics. Chapter four covers the performance analysis of the protocol including timeline traces of function calls on the end systems and network analyser traces captured from the transmission medium. The delay results under normal operation and under session redirection are presented and discussed. Chapter five draws conclusions from the previous chapters results and summarises the work. A comparison is made to the previous work identified in chapter two and ideas for future directions of work are offered. Finally the appendices contain the initial project proposal along with the references and bibliography. A full copy of all commented source code is included at the back on a DOS formatted floppy disk.

Chapter 2 - Background

This chapter introduces the relevant background information that is necessary for reading the rest of this document. First of all the structure of an abstract cellular network is introduced followed by a discussion of IP and Mobile IP routing and how they can be mapped onto the cellular architecture. Subsequently there is a literature review that discusses and evaluates other researcher's work in the same area as this. Finally the multi-homed management protocol is discussed as a constituent part of a framework for integrated mobile Quality of Service (QoS), security and billing.

2.1 Cellular Network Structure

The first chapter mentioned that it is widely thought that future cellular networks will be based completely on Internet Protocol approaches [Nokia99]. This means that each device in the cellular network will have its own IP address and IP routing will be used to transfer data packets from sender to receiver. Current cellular networks such as the Global System for Mobile communications (GSM) standard [Scourias97] do not use IP as a network protocol. In fact GSM is a circuit-switched technology, which requires an end-to-end connection to be established before any data or voice information can be transmitted. This differs wildly from the packet-switched world of IP where, at the network level, no end-to-end connection exists and individual packets can travel through different routes to reach their destination. Currently, in order to transport IP packets on top of GSM an appropriate link layer protocol – such as Point-to-Point Protocol (PPP) [Simpson93] or Serial Line IP (SLIP) [Romkey98] is required. These two protocols are more commonly used for home users to dial up to the Internet across a fixed telephone line using a modem (that is what is happening here except that the fixed line is substituted by a mobile telephone line). It is important to notice, however, that using PPP or SLIP over GSM is still using the circuit-switched paradigm. Because packet-switched networks – such as the Internet – do not require an end-to-end connection, network resources and bandwidth are only used when data is actually being transmitted. In a wireless environment this means that the available spectrum can be shared efficiently between users. So how will an "all IP" cellular network differ from current cellular networks such as GSM?

To describe an IP cellular network it is necessary to first describe the components of any cellular network in an abstract way. First and foremost cellular networks get their name because the coverage area is split into a number of regions called cells. The reason for this is so that frequencies can be re-used in non-adjoining cells.

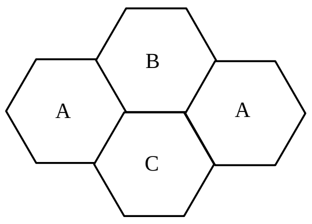

Figure 1 – Frequency Re-use in Cellular Networks

Figure 1 shows four cells in a cellular network with the frequencies shown as capital letters inside each cell. The rightmost and leftmost cells are able to use the same frequency (A) because they are non-adjacent. A Base Transceiver Station (BTS) controls the communication within a cell. The functionality of a BTS is kept to a minimum and normally involves transmission and reception, signalling and interaction with other network elements. A Base Station Controller (BSC) is a device that coordinates the actions of multiple (normally tens of) BTSs. The BSC contains more intelligent circuitry and controls mobility functions such as the movement of a mobile device between cells and hence between BTSs. This process of movement between cells is known as hand-off. Hand-off between BTSs that are under the organisation of different BSCs is controlled by a network element known as an MSC (Mobile Switching Centre). The MSC is also responsible for linking the cellular network with the outside world – Public Switched Telephone Networks (PSTNs) and other cellular networks. Figure 2 (from [Hamalainen96]) shows the architecture of a cellular network.

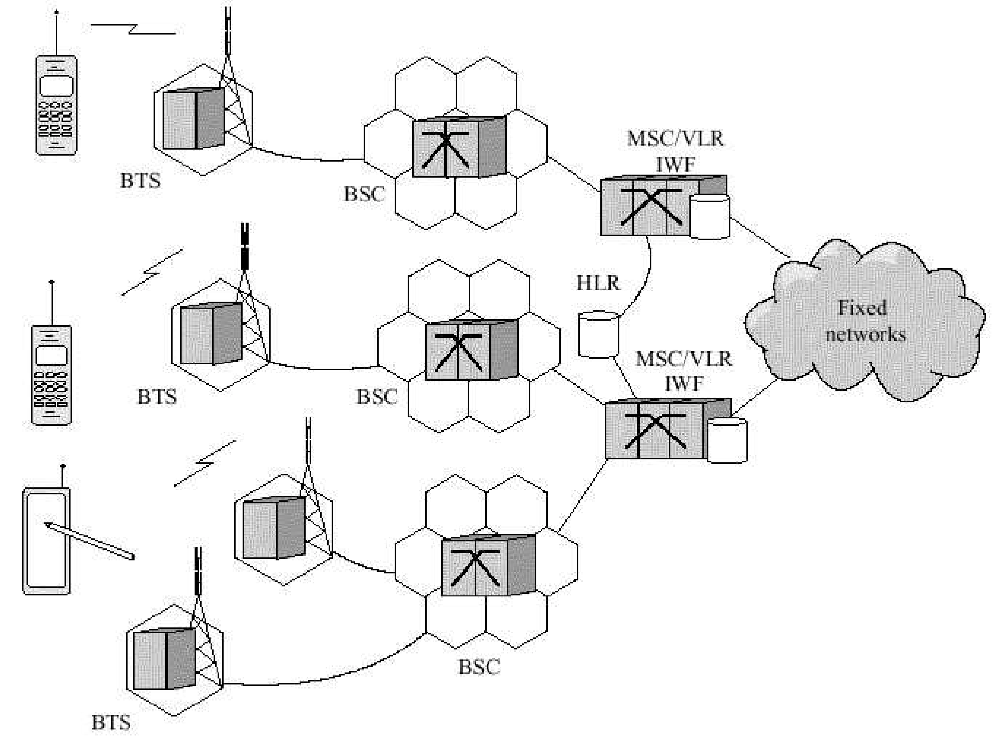

Figure 2 – Example Cellular Network Structure (from [Hamalainen96])

Now that there is a reference model for an abstract cellular network the next three subsections describe how IP methods can be mapped onto this framework.

2.2 IP Network Architecture, Addressing and Routing

In an IP network there are end systems known as Hosts and intermediate systems called Routers. Hosts communicate with each other by the transmission of IP packets called datagrams. If two hosts are not connected to the same physical medium – known as a subnetwork – then datagrams must be forwarded to the destination subnetwork through one or more routers. Figure 3 shows an example IP network where datagrams from the two hosts may travel through a number of different intermediate routes.

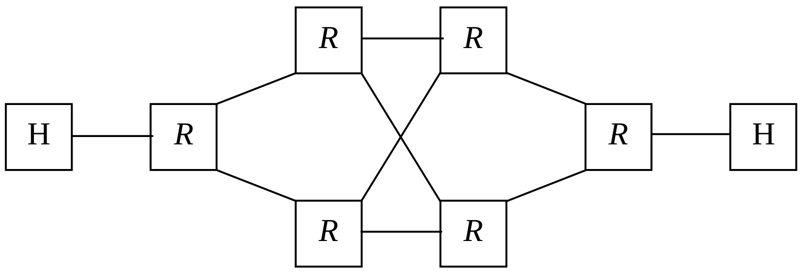

Figure 3 - Example IP Network Architecture

Each element of the IP (version 4) architecture is given a 32-bit IP address (although this is set to change to a 64-bit address with IPv6 [Huitema97]). This means that hosts, routers and networks are all given an address. As it is important to understand the subject of IP addressing and routing before reading about the multi-homed management protocol designed in this work the following paragraph gives an overview of the subject. Further information on IP addressing and routing can be found in the book "TCP/IP Illustrated Vol. 1" by Richard W. Stevens (see Appendix B - Bibliography).

Although computers see the IP address as a series of 32 bits it is usually quoted in dotted decimal form to make it more readable to the human eye. Dotted decimal IP addresses are written in the form `aaa.bbb.ccc.ddd.` The address can be converted into a 32-bit binary number by converting the individual components (`aaa`, `bbb`, `ccc` and `ddd`) into binary form and then simply concatenating the separate strings into a 32-bit number. A subnet mask is also associated with each host and router in order to determine the address of the network to which they belong. The subnet mask is again a 32-bit number also quoted in dotted decimal or hexadecimal form. The subnet mask consists of ones corresponding to the binary columns where the network address lies followed by a series of zeros to fill the 32-bit number. An example is the host 143.167.11.192 with subnet mask 255.255.252.0. First the dotted decimal formats must be converted to binary strings:

```
143.167.11.192 equals 10001111101001110000101111000000 in binary
255.255.252.0  equals 11111111111111111111110000000000 in binary
```

The network address of the host is calculated by the logical AND of the two bit patterns:

```
(143.167.11.192) 10001111101001110000101111000000 AND
(255.255.252.0)  11111111111111111111110000000000 =
(143.167.8.0)    10001111101001110000100000000000
```

So the host 143.167.11.192 with subnet mask is connected to the network with address 143.167.8.0.

IP routing requires a router to be able to determine the address of each of the networks to which it is connected so that the router can calculate whether a destination host's address is on one of the networks or not. If the destination address is on a connected network then the router can send the datagrams to the destination, otherwise it must forward the datagrams to another router.

2.3 Mobile IP Routing

IP routing is further complicated when hosts are mobile and may constantly change their point of attachment to the Internet. For the delivery of datagrams to a host to work properly an assumption is made that the host is always located on the network indicated by the host's IP address. If a host changes its point of attachment to the Internet then the address of the network to which the host is connected also changes. This means that the whole IP addressing and routing scheme no longer works: a host may not always be connected to the network whose address is determined by the logical AND of the host's IP address and subnet mask pair. Mobile IP [Perkins96] solves the problem by giving a mobile host a temporary address on each visited network. The mobile host always has an IP address for its home network but when visiting a foreign network it is assigned a foreign IP address – known as a care-of address. Datagram delivery to the mobile host visiting a foreign network is achieved by tunnelling IP datagrams from the home network to the foreign network using the care-of address as the destination. Tunnelling is the process of encapsulating each IP packet sent to the host's home address within another IP packet that is sent to the host's care-of address. Two new entities are introduced to accomplish this. A home agent is a router on the mobile host's home network that can intercept packets destined to the mobile host's home IP address and tunnel them to the host's current care-of address. A foreign agent is a router on the visited network that can de-tunnel datagrams for delivery to the mobile host. The architecture is shown in figure 4.

In order to publicise its services and so that a mobile host can determine when it has moved the foreign agent periodically transmits advertisements. These advertisements are in the form of an extension to the conventional IP router advertisement messages [Postel81b]. When a mobile host obtains a new care-of address it must register this address with the home agent so that datagrams can be tunnelled to the correct address. This is known as registration.

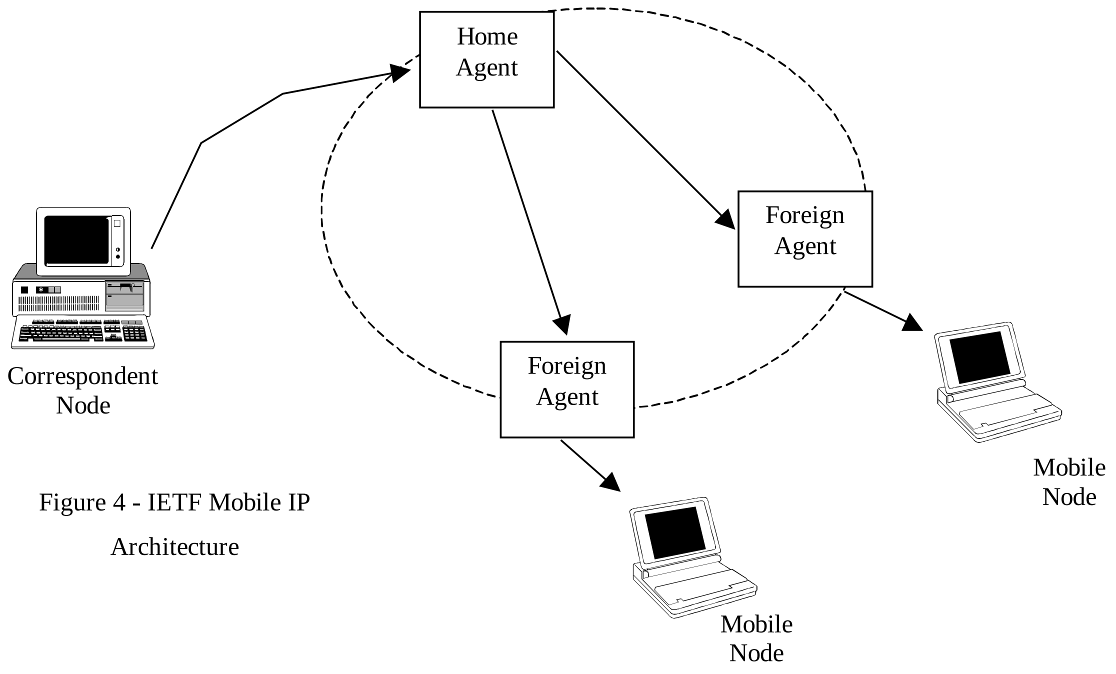

Figure 4 – The IETF Mobile IP Architecture

Communication between a mobile host and a correspondent host takes place as follows. The correspondent sends a datagram to the mobile's home address. The home agent on the mobile's home network intercepts the datagram through the use of proxy ARP [Postel84]. The home agent tunnels the IP datagram to the foreign agent on the visited network, which then de-tunnels the datagram and delivers it to the mobile host. The mobile can send datagrams to a correspondent using its home IP address as the source address in the IP packet. However, for security, some firewalls filter out datagrams that contain source addresses that do not belong to the network from which they originate – this is called ingress filtering. A solution to this problem is for the foreign agent to reverse tunnel datagrams originating from the mobile back to the home agent for transmission.

Mobile IP is not without problems, one of which is known as triangular routing. This means that no matter where the mobile host is, packets have to be routed through the home agent and back to the correspondent node. This is particularly troublesome if the mobile is close to the correspondent but a long way from the home agent. An example of this is a user whose home network is in England but they are roaming in Japan. Although they may be communicating with another user in Japan, all packets are transported via England. This is further complicated if a foreign agent is used (quadrilateral routing).

2.4 IP based Cellular Networks

It is now possible to describe how the IP practices described above could possibly be mapped onto the abstract cellular network architecture described in section 2.1 above. It must be noted however that this description is only an example and the actual implementation is left as a design decision to each network operator. In the cellular architecture there are Base Transceiver Stations (BTSs), Base Station Controllers (BSCs) and Mobile Switching Centres (MSCs). In the IP world there are Hosts, Networks and Routers along with Home and Foreign Agents for Mobile IP (HAs and FAs). It is obvious to see that mobile devices in the cellular environment will be equivalent to IP hosts. The mapping of IP subnetworks onto cells leads to more than one choice. One possibility is to let the coverage area of one cell be equivalent to one IP subnetwork. In this case each BTS would have to contain the functionality of a router, which would mean an increase in the complexity of the circuitry and the overall cost of maintaining and installing the network. It is more likely that an entity further up in the hierarchy such as a BSC or MSC is given the task of being a router. Each subnetwork further requires a Mobile IP HA and FA, although these two devices could be combined with each other and possibly together with the router. Figure 5 shows the architecture of the IP based cellular network.

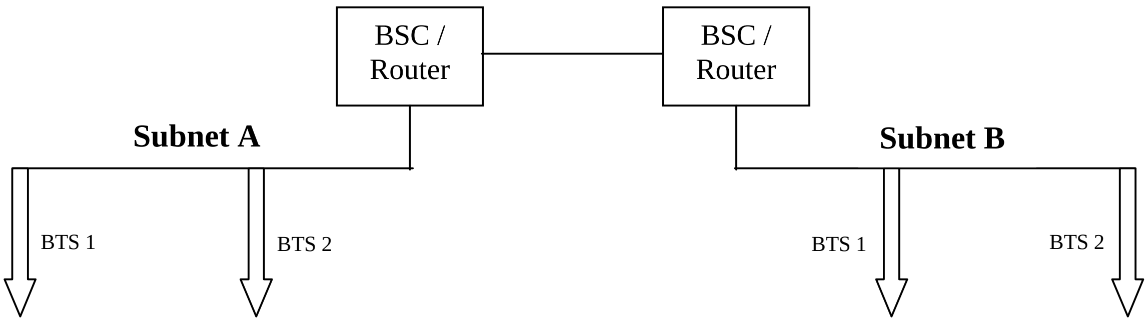

Figure 5 - Architecture of an IP based cellular network

When a mobile device moves from the cell controlled by BTS1 to that of BTS2 in Subnet A there is a link-layer handoff but the device remains in the same IP subnet. However, when the mobile moves from the cell of BTS 2 in subnet A to the cell of BTS1 in subnet B there must be a link-layer handoff and also a subnetwork handoff. The former is known as micro-mobility and the latter as macro-mobility. Mobile IP is used to control the macro-mobility handoff hence the two elements named “BSC / Router” in the diagram also act as foreign agents so that the mobile can obtain a new IP care-of address when moving between subnets.

2.5 Literature Review

The next six subsections describe the work of other researchers in a similar area to the multi-homed management protocol. For each one the subject is described first, followed by a summary of the main findings and a discussion of the differences and similarities to this work.

2.5.1 Flexible Network Support for Mobility

The work of Zhao et al [Zhoa98] focussed on the management of multiple connections at the network layer. The work provided two extensions to Mobile IP to provide multiple packet delivery mechanisms and to utilise multiple active network interfaces. It was suggested that the ability to send multiple packet types is needed because Mobile IP may not be required for certain types of traffic. For instance, web-browsing connections are short-lived and therefore it is not very likely that hand-off will occur during the lifetime of a connection. The need to support multiple interfaces was argued in terms of making routing decisions based on the characteristics of the traffic: QoS, security, cost and billing; but also to enable smooth hand-offs and for link asymmetry. (Link asymmetry refers to the case where the connectivity is only in one direction, such as in a satellite system where traffic can flow in the downstream direction only). The authors identify the fact that using multiple interfaces for transmission is the simple case and reception poses a more difficult problem. To solve this problem they propose Flow-to-Interface bindings as an extension to the standard Mobile IP registration packets with multiple tunnels from the home agent to the mobile node. The implementation made changes to the Linux kernel and provided two new socket options so that mobile aware applications could use the added features. The route lookup function in the kernel was modified to take into account traffic-flow characteristics – UDP and TCP port numbers – when making routing decisions.

Experimental work showed that the process of flow de-multiplexing at the Home Agent caused latency from 2.3 microseconds for one Flow-to-Interface binding to 9.2 microseconds for 60 bindings. It was noted that although this delay may seem large compared to normal Mobile IP de-multiplexing on the Home Agent (2.1 microseconds in these experiments) it was, however, a small cost compared with the round trip delay between the mobile host and the correspondent host.

Zhao's work differs from this one in many ways. Firstly the work depends on the Mobile IP architecture, whereas the protocol described in Chapter Three of this document does not rely on any single mobility protocol and can be used with any underlying network protocol: IP, Mobile IPv4 or Mobile IPv6. Furthermore the modifications to the Linux kernel described above mean that existing applications have to be modified to take advantage of the new functions. One of the primary aims of the work described in this thesis is that legacy applications will work with the multi-homed management protocol transparently without the need for modification. Finally Zhao's work has a scalability problem. Because the port number is used to make routing decisions IP packets that become fragmented must be reassembled at the home agent before being transmitted to the mobile node. This exasperates the problem of triangular routing especially in the extreme case – when the mobile and correspondent are close to each other and the home agent is far away.

2.5.2 Wireless Overlay Networks

The term 'wireless overlay network' was first introduced in [Katz96]. A wireless overlay network consists of several tiers of communications technology each with different characteristics. At the top is a network that covers a wide geographical area but provides a small amount of bandwidth per unit area: such as a UMTS satellite or cellular network [UMTSForum99]. The middle tier is a metropolitan area network that provides higher bandwidth, but limited geographical coverage, for instance wireless technology such as IEEE 802.11 [IEEE99] or HiPerLAN [ETSI98]. Finally, the lowest tier is a local area network that provides very high bandwidth to the user, but has a small coverage area. Typically the coverage of this bottom layer of the overlay is limited to a single room using technology such as Bluetooth [Bluetooth99] or IrDA [IrDA96]. The work described in [Katz96] concentrates on the interaction and handoffs of the mobile host between the heterogeneous wireless networks. The emphasis is on 'vertical handoff' – handoff from a cell in one wireless network to a cell in a different wireless network rather that 'horizontal handoff' – movement between cells in the same wireless network.

The performance of vertical handoff in the wireless overlay networks described above is presented by Stemm [Stemm98]. A version of Mobile IP is adapted so that the mobile's care-of address is multicast. Base stations acting as foreign agents are selected by the mobile to listen to the multicast address for packets that are being tunnelled to the mobile from the home agent. One base station is selected to decapsulate and forward packets to the mobile, while the other base stations cache the packets in a buffer. When the mobile does a handoff it signals the forwarding base station to go into buffering mode and the new base station to start forwarding packets.

The results show that in the best case, handoff from a room-size network (infrared) to a building-size network (IEEE 802.11) took 170ms and cost 1.5% overhead in network resources. Hand-off between a building-size network and a wide-area network (Metricom Ricochet) took 800ms and cost a similar overhead in resources.

Stemm's work differs from this one in that changes are made to the wireless network. This work only makes changes to the network stacks on the hosts – there is no need to change the structure of the network. Again the work of Stemm is reliant on a Mobile IP architecture whereas this work is abstract from the mobility protocol used and hence will work with future IP mobility protocols.

2.5.3 MSOCKS: An Architecture for Transport Layer Mobility

Transport layer management of multiple connections is analysed in [Maltz98]. The work again emphasises the belief that routing using multiple network interfaces should be dependent on the link characteristics that best match the application's needs. The work uses a proxy server that manages the TCP connection or a series of UDP datagrams between the mobile host and the correspondent node. The proxy handles the redirection of streams when the mobile node changes its network interface and hence IP address. For TCP a technique known as TCP Splice is used so that the TCP connection split over multiple streams has the same operation (flow control and error handling) as that of a normal TCP connection. The performance of the architecture was analysed in terms of the number of simultaneous mobile hosts the proxy server could support. In addition to this the latency added to the communication path by the proxy was determined. The results show that the proxy is scalable by displaying a drop in throughput from approximately 90 to 70 MBps as the number of connections is increased from 0 to 250. The latency involved with the connection redirection is 10% compared to that of normal IP forwarding.

The work on “Transport Layer Mobility” differs in approach from the work described here. The use of a proxy server to control the switching of data flows between wireless connections is an alternate method to the work presented here, which uses signalling between end-systems to achieve redirection of communications sessions.

2.5.4 Handoff in Hybrid Mobile Data Networks

Hand-off between an IEEE 802.11 and a GPRS network is examined in [Pahlavan99]. The work presents an analytical view of a scenario where a mobile user is travelling in a straight line away from the 802.11 access point and has to hand-off to the GPRS network, which overlaps the 802.11 cells. This work is similar to that of Stemm [Stemm98] although an analytical approach is taken to model the handoff between the different wireless networks as opposed to empirical testing. The results from simulation show that greater throughput is achieved if hand-off is left until the last possible minute allowing the user to use the higher bandwidth of the 802.11 network as far as possible.

2.5.5 Dynamic Network Reconfiguration Support for Mobile Computers

Physical Media Independence (PMI) is an architecture described in [Inouye97] that allows for the dynamic reconfiguration of network interfaces such as PCMCIA and FireWire that can be added or removed whilst the system is running. The PMI system detects the available devices connected to the mobile node and uses meta-data to inform all of the layers in the network stack, so that each layer can make the adaptation best suited to itself. It is shown that certain applications, such as streaming video, make assumptions about the state of the underlying network and should therefore be notified when the interface used for network connection changes.

This system breaks the mould of networking protocols in communicating with each layer rather than the norm of abstraction between individual layers. The work presented in this thesis does not take this path – the session layer protocol works independently of the rest of the network stack, enabling it to be used on a much wider range of existing systems.

2.5.6 Stream Control Transmission Protocol

The mission of the Internet Engineering Task Force's SIGTRAN working group concerns the transmission of signalling information. A recently designed protocol called Stream Control Transmission Protocol (SCTP) [Stewart00] allows signalling data for multimedia to be carried on top of IP. The SCTP protocol provides a means to deliver user data through multiple streams and to send multiple user messages through a single stream. The protocol also provides multi-homing for fault-tolerance purposes. It is important to mention at this juncture that the SCTP protocol is not a mobility protocol and has been designed for static Internet hosts. However, the way in which it directs data streams through multiple network connections is of interest with respect to this work. SCTP works in a connection-oriented manner: an association specifying the available IP addresses and SCTP ports on each host must be established before any data transfer is possible (although user data can be piggybacked during the last two stages of the four way handshake).

SCTP streams are defined as sequences of user messages rather than the TCP definition of a stream as a sequence of bytes. The reason for this is so that the current TCP method requires that applications must differentiate consecutive messages in the byte stream and must ensure that whole messages are transmitted at once. SCTP supports multi-homing for fault-tolerance. A stream is always assigned to the primary path (i.e. the default network interface) until a retransmission is required. If data is lost and needs to be retransmitted it is sent via the combination of local interface and destination interface that differs the most from the original. This attempts to ensure that network errors at either end-point are avoided.

The SCTP protocol is designed as a replacement for TCP and as such it is very different from the multi-homed management protocol. SCTP provides its own acknowledgements, flow-control and sequencing. Chapter three of this document explains how the multi-homed management protocol is intended to be used with the existing Internet transport protocols UDP and TCP. Furthermore the use of multi-homing in SCTP is different from its intended use here. In this work the idea is to transmit different streams of data across different paths in order to increase the bandwidth available to the mobile host. In addition SCTP provides no mobility functions and so the redirection of streams to different interfaces is only included for fault tolerance and not for handoff purposes.

2.6 An Integrated Architecture for Mobile Quality of Service, Security and Billing

The literature covered in the previous subsection has a common theme. Current routing decisions on mobile hosts are made solely on the destination address of the IP packets and do not take into account other characteristics of the traffic. The common belief in all the literature reviewed is that multi-homed mobile hosts should be able to make routing decisions based on other factors such as Quality of Service (QoS), security and billing. It should be possible for the user or an aware application to set the various importance of each characteristic for each communications session. Based on these settings a management layer can make an informed decision about which interface to use for a specific session. For instance, if a mobile device has a UMTS link with 2Mbps and a Bluetooth link at 421kbps the user could define to use the Bluetooth link for email because email is not delay or bandwidth sensitive and the Bluetooth link is free. However, if the user were to hold a videoconferencing session the UMTS link could be chosen because it provides higher bandwidth and therefore will result in a better overall QoS. Security can be looked at in the same way: surfing the web for leisure has different security requirements than for e-commerce – hence this influences the selection of which interface to use. Handoffs at the link-layer will raise the need for session redirection: when the user moves out of the coverage range of the Bluetooth link email traffic will have to be redirected through the UMTS link.

It is intended that the multi-homed management protocol handle these two functions of initial interface selection based on user defined parameters and session redirection after hand-off. The design and implementation of this protocol are discussed in the following chapter.

Chapter 3 - Protocol Design and Implementation

This chapter describes in detail the protocol functionality. First of all the requirements for the operation of the protocol are presented. Subsequently, the design of the protocol is discussed. This section covers the different options available for the design and a justification for each choice made. The final section gives details on how the protocol was implemented; again with the different possibilities and the justification for the chosen implementation.

3.1 Requirements of the Protocol

A broad statement of requirements for the protocol is: "Assign each newly initialised communication session to an appropriate interface and redirect the communication session to a new interface when the current one goes through a period of handoff". This statement consists of two halves. Firstly, when a new communication session is started the protocol should decide upon an appropriate interface to use for that particular session. This is the first requirement of the protocol and from here on it will be referred to as *interface selection*. Secondly, when an interface has to move between subnets – and consequently there is a handoff period where data-loss could occur – the protocol must be able to redirect any session bound to that interface through a suitable alternative. This is the second requirement of the protocol and will be referred to as *session redirection*. It is important to note this requirement handles handoff between subnets (macro-mobility) and not handoff between cells within the same subnet (micro-mobility). Micro-mobility is, at present, left to the appropriate link-layer technology and is not part of the protocol requirements. The reason for this is that the scope of this work is to enable a mobile host to connect to and to communicate within multiple heterogeneous wireless networks. Improving micro-mobility handoff is not within this scope. Moreover, it could be argued that the individual link-layer technologies are better at micro-mobility handoff than a protocol such as this which operates so high up in the network stack. The final requirement is that the protocol should operate transparently to the application layer so that legacy applications can be supported without modification.

Now that the two main requirements of the protocol have been put forward the specific requirements of each can be discussed.

3.1.1 Interface Selection Requirements

Upon an application request to initialise a new communication session with a correspondent host the protocol must select the appropriate interface for the session based upon the traffic characteristics of the session and user choice. This responsibility of the protocol is the part that interacts with the architecture described in section 2.6 above. The task of interface selection is to take user-defined settings for desired level of QoS, security and cost and return the interface that provides the best correlation.

The interface selection operation should be able to read user-defined and application-defined parameters from a database along with physical attributes of the available interfaces from the operating system kernel. When it has this information, the interface selection operation should make the appropriate decision based on some heuristic or algorithm known as the interface selection algorithm.

3.1.2 Session Redirection Requirements

The protocol must detect when an interface moves between subnets and redirect any active sessions bound to this interface through an appropriate alternative. A message should be sent to the peer to notify them of the change of interface. This is called a *binding update*. The alternative interface must be selected through the use of the interface selection algorithm.

The detection of handoff is an important issue. If handoff can be signalled pre-emptively through a link-layer hint the session redirection can take place before handoff and therefore no data-loss will be incurred. Monitoring the signal strength and issuing the hint when it drops below a certain threshold could achieve this. This would, however, rely on the modification of each link-layer device driver on each operating system for each type of wireless technology supplied by each vendor and is as such not currently plausible. However, it is interesting to include the future possibility of having such a link-layer hint and as a consequence this has been incorporated in the protocol design (see section 3.2 below). For this purpose it is necessary to go into a little more detail on current link-layer handoff decisions in order to describe how the hint could be given. In an abstract cellular network architecture (section 2.1 above) the mobile device constantly senses the signal strengths from the available BTSs and selects one to bind to. A decision to bind to a new BTS’s signal – and hence handoff – is made through an algorithm or heuristic (although the network can actually move mobiles between cells for load-balancing reasons).

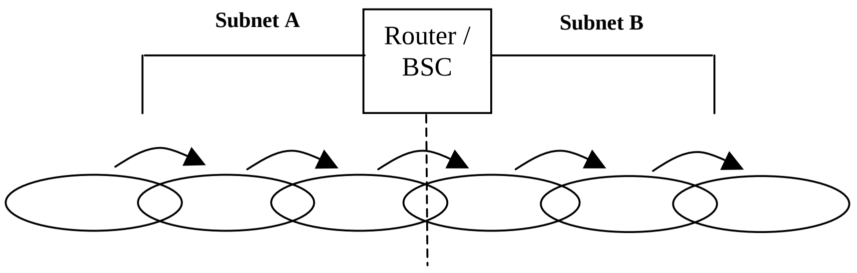

Figure 6 - Movement in an IP Cellular Network

Figure 6 shows a simplified view of how a mobile device moves between subnets in an IP cellular network. There are two subnets (A and B) in the diagram, connected by a router. The cells within each subnet are shown as ellipses and the individual link-layer handoffs are shown by arrows. The dotted line shows where the boundary between cells in different subnets lies and hence the arrow designating movement across this line shows a subnet movement (macro-mobility). Each cell in the architecture is given a unique identifier, which could be linked to subnetwork address. For instance reading from left to right in the diagram the cells could be called A1, A2, A3, B1, B2 and B3. When the mobile device reaches the subnet boundary it realises that (from the cell identifiers) it is about to move between subnets hence it issues a pre-emptive hint of macro-mobility to the multi-homed management protocol in the session layer. The process of session redirection can then begin.

The above paragraph is the *desired* operation of link-layer hint where the mobile can predict subnet movement in enough time for session redirection. However, if this is not possible the only method to detect handoff is to be notified when an interface goes down and issue the session redirection message (binding update) through an alternate interface. In this case the latency caused by session redirection is critical since during this period any user data transmitted will be lost. Consequently it is imperative that session redirection take the minimum amount of time. Moreover, since the session redirection operation also uses the interface selection algorithm the efficacy of the whole system is hinged upon the latency due to these two functions. A further complication is that of security. If a binding update for a session can be issued by the new interface instead of the previous one, then the matter of authentication is imperative. However, the security of the protocol is out of the scope for this work and is discussed as a part of the future work needed (see section 5.2 - Future work).

3.2 Protocol Design

The last section outlined the requirements for the protocol and refined them into two particular components. This section will take these requirements and produce a design for the operation of the protocol. Where design decisions have to be made they will be specifically noted, explained and justified.

3.2.1 Placement in the Network Stack

The first design decision is at what layer in the network stack to host the protocol. It has already been stated in this document that the protocol operates at the session layer, but why? Firstly, the session layer of the Open Systems Interconnection Reference Model (OSI/RM) [ISO84] deals with streams of data called sessions. A session can be defined as logical sequence of communication between two applications [Halsall96]. A typical application – such as videoconferencing – may have multiple sessions (streams of data): audio, video and text transcription in both directions as well as signalling information. These sessions have varying levels of QoS and security requirements and hence it is desirable to make interface selection decisions on a per-session basis rather than a per-packet basis (in the case where the protocol would operate at the network layer).

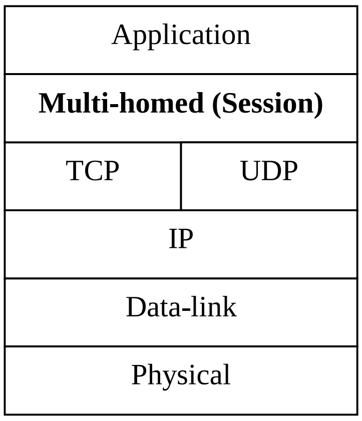

Figure 7 – Location of the Protocol in the Network Stack

Since the protocol is to be located in the session layer this means that it will operate above the Internet transport protocols – UDP and TCP – and below the application layer (see figure 7). UDP (User Datagram Protocol) [Postel80] provides a best-effort connection-less datagram transport. This means that the protocol merely provides an interface to the IP layer and does not provide any flow control or error-correction. TCP (Transmission Control Protocol) [Postel81c] is a connection-oriented, reliable transport protocol that supplies flow-control and error detection and retransmission. The requirements for the multi-homed management protocol (section 3.1 above) clearly state that the protocol must operate transparently to the application layer so that legacy applications are supported without modification. Accordingly, the interface that this protocol provides to the application layer must be the same as that provided by UDP and TCP. Hence, the protocol must provide two types of functionality – one for when the application requires a UDP association (connection-less, unreliable) and one for TCP associations (connection-oriented, reliable). These two sets of functionality differ in the way that signalling information is transmitted.

3.2.2 Signalling

Any protocol needs a way of communicating information relating to protocol operation rather than the actual data to be transported by the protocol. This information is known as signalling. The most important aspect of signalling related to this work is to decide how the protocol creates an association with its peer protocol on the correspondent host and then how it signals session redirection to the peer. It is important at this point to differentiate between the terms *connection* and *association*. An association in terms of a connection-oriented protocol is a logical connection that is set-up through the use of signalling before any data can be sent. In the case of TCP the signalling is called a three-way handshake: one peer initiates the connection set-up by issuing a connection request, the opposite peer then replies with a connection response that can either be negative or positive and finally the first peer acknowledges the response. It is not until after the final acknowledgement that the connection is established and data can be sent between the peers (assuming a positive response). The two peers are then said to have an association between them. By contrast, UDP is a connection-less protocol and data can be sent immediately without the need for the three-way handshake. However, two UDP end-points still need to have some association with each other that is created when the server first receives data from the client. An association is named at each end by the pair of IP address and TCP or UDP port number on either host.

As stated in section 3.2.1 above the multi-homed management protocol must work on top of UDP or TCP and must provide the corresponding functionality to the application. However, the multi-homed management protocol also needs to transmit its own signalling data. This signalling data is necessary for the session redirection function, where the peer must be told that the session will be moved to a different interface. Furthermore, since the IP address of either client or server could change at any point, due to mobility, a more specific way to name an association is required. The proposed solution is to use session identifiers. In this way, each session initiated is given a number that is unique to the host. When an association is created the session identifiers are exchanged and then if session redirection is needed the two peers have a way to identify the session once either IP address changes.

There are two choices of how to transport signalling data. The first is to have a specific port on each machine that listens for signalling data concerning the protocol. The second option is to include signalling information inline with the data-flow. To accomplish this a session layer header must be prefixed to each application layer protocol data unit (PDU) before passing it on to the transport layer. The latter has been chosen to enable a session identifier to be associated with each PDU transmitted. If the former were used this would be impossible and there would have to be an alternate method of identification.

Now that the transport of signalling information has been discussed it is possible to design the actual signalling mechanisms and information in more detail.

3.2.3 Interface Selection

The interface selection algorithm is a key feature of the protocol design. The algorithm must take an application's requirements for type of service and match them to the user's constraints for type of service. This battle between the resources that an application wants and those that the user is prepared to allow it to take must also be matched with the demands of other applications running on the system. The important aspect of this is that the interface selection algorithm should be a component that can be substituted so that different algorithms can be designed and tested. This would lead to another research area (see section 5.2 - Future work) to decided upon an appropriate algorithm. In terms of software engineering the design of the protocol must be abstract from the particular interface selection algorithm. There are two functional elements to interface selection. The first is to enumerate the interfaces that are available at the time that the session is first initialised. Secondly, based upon the results of the first part an appropriate interface should be selected according to the interface selection algorithm.

When an interface for the session has been chosen an identifier is assigned to the session and data can be sent out of the appropriate interface. In order to signal the session identifier (SID) to the correspondent host a session header is needed. The proposed session header structure is shown in figure 8.

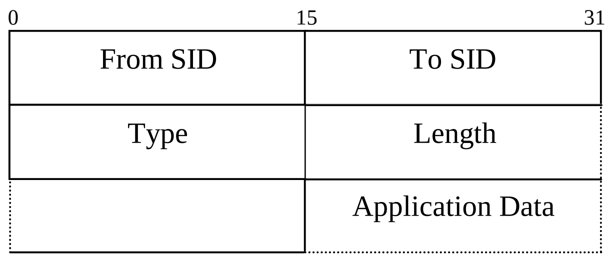

Figure 8 – Session Header Structure

The session identifiers are unsigned 16-bit integers. The reason for this is that the range for UDP and TCP port numbers is 0 through 2^{16} and therefore the maximum number of sessions on a host is 2^{16}. The type field is used to denote whether the PDU is a normal data packet or a binding update packet (session redirection). The value 01 has been chosen arbitrarily for the normal data transfer type and the value 05 for the binding update type. The length field is a 32-bit integer, which identifies the length of the application data contained in the PDU. The size of this field introduces a limit to the maximum application data size of 2^{32} = 4294967296 bytes (= 4GB), which is considered sufficient both at the present time and for the immediate future.

The next stage in the design is to decide how session identifiers are exchanged between two peers. The session layer protocol is not connection-oriented in a strict sense of the term because it must provide a transparent interface for the application to use UDP. However, there must be an association between the two peers for session redirection. This is achieved as follows. The From SID is inserted into the initial PDU and the To SID is left blank (the type and length fields are assigned their corresponding values). When the correspondent receives the PDU it must store the From SID in order to identify the sender. When the correspondent replies to the sender it creates a header containing the SID received from the sender in the To SID field and inserts its own SID in the From SID field. The result of this is that if communication is in one direction only, the server will never transmit and therefore will not send its SID to the client.

3.2.4 Session Redirection

Section 3.1.2 above states that there are two possibilities for detection of movement between subnets. In the first case the link-layer provides a hint to the session layer protocol to indicate that an interface is about to move to a new subnet. The second case deals with the occasion where no hint is available or the hint is not received in time. The operation of the protocol and the associated signalling must now be defined. In either case the session redirection is achieved by sending a binding update message to the peer host. This is achieved by sending a session layer PDU of type 05. In order to notify the peer of the IP address of the interface that will be used to continue the session an extension of the header shown in figure 9 is used.

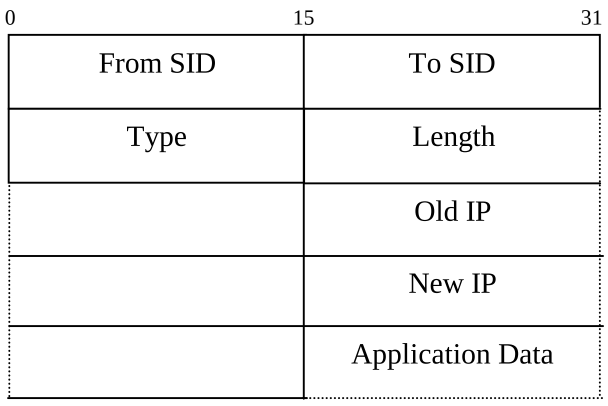

Figure 9 – Binding update request Header Structure

The binding update request header consists of the same four fields as the normal session layer header (figure 8): From SID, To SID, Type and Length; along with two new fields to denote the IP address of the interface that the session was first bound to (Old IP) and the IP address of the interface to move the session to (New IP).

When a link-layer hint is received a binding update message is transmitted through the interface that is about to hand-off. However, if the link-layer goes down before the message can be transmitted then the binding update must be retransmitted through the new interface. When a host receives a binding update message it must first verify that it has a SID-to-interface binding for that particular correspondent interface IP and SID. If so then this binding must be amended to use the new IP address as destination for all future data. The details of how this is achieved with UDP and TCP are given in sections 3.2.5 and 3.2.6 below. Figure 10 shows a timing diagram for protocol signalling of interface selection and session redirection information.

The first part of the figure shows the session association establishment signalling. The client sends an association request contained in the session layer header described above. The server transmits an acknowledgement to this request containing its own SID. Either side stores its own session-to-interface binding as well as the peer's session-to-interface binding. It must be noted that these bindings are one-to-one and hence at present the protocol does not support multicast communication. An explanation of possible operation with multicast is given in section 3.2.8 below. The second part of the diagram (after the series of dashes in the time-line) shows the binding update signalling. First a binding update request is sent (see figure 9). The peer host replies with a binding update response that is different depending on whether the session is UDP- or TCP-based. Finally a new association request and acknowledgement must be exchanged before the association is considered re-enabled. The differences between UDP and TCP binding updates are discussed below.

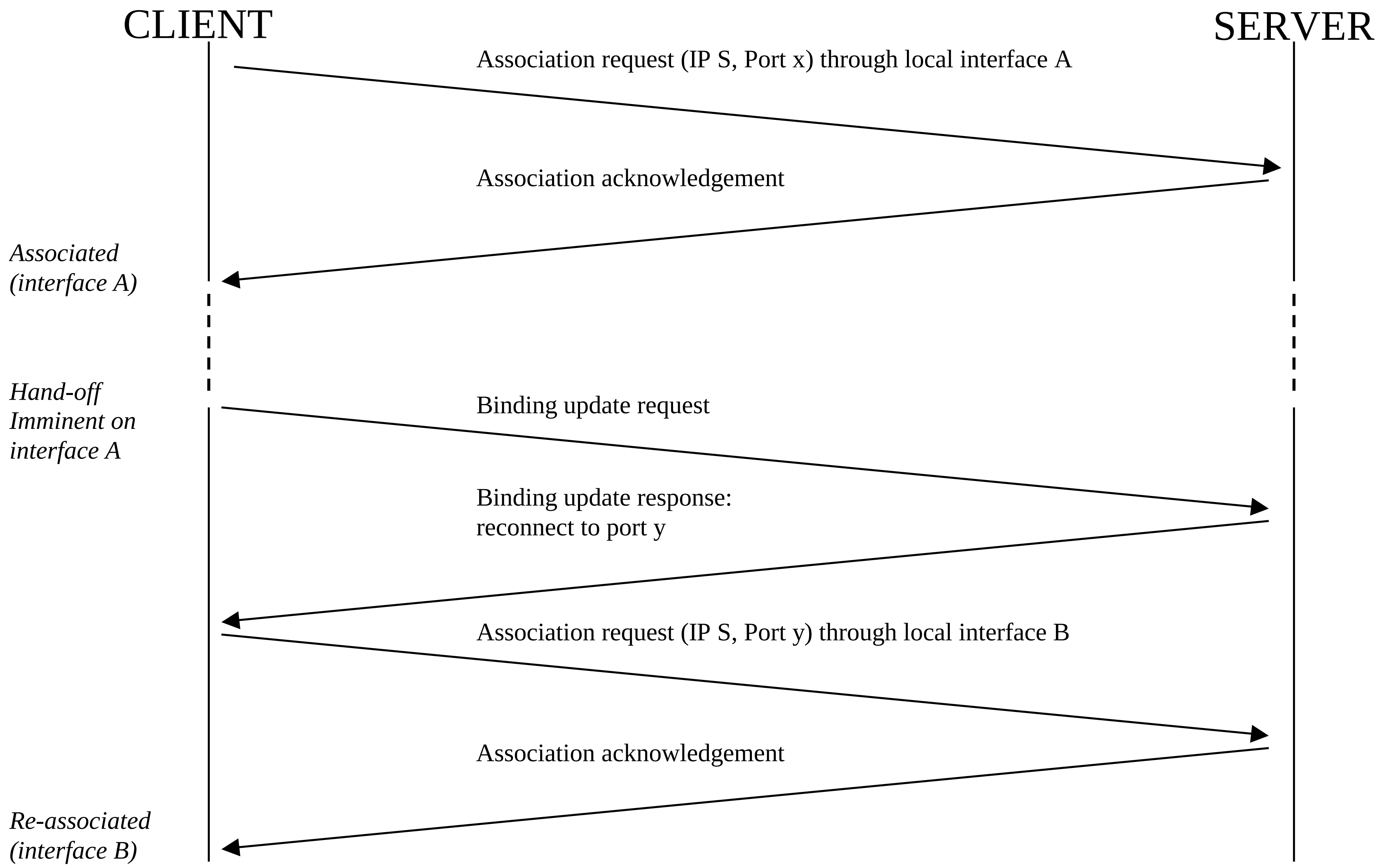
CLIENT
SERVER
Association request (IP S, Port x) through local interface A
Association acknowledgement
Associated (interface A)
Hand-off Imminent on interface A
Binding update request
Binding update response: reconnect to port y
Association request (IP S, Port y) through local interface B
Association acknowledgement
Re-associated (interface B)

Figure 10 – Session Layer Signalling

3.2.5 Operation with UDP

The UDP binding update process is very simple. The mobile issues a binding update request that informs the peer that the session is to be continued through the new interface with a given IP. The peer merely acknowledges this and the session continues through the new interface.

3.2.6 Operation with TCP

Due to its connection-oriented nature TCP's binding updates are more complicated. The existing connection must be closed and a new connection established through the new interface. In order to achieve this the peer must notify the mobile of the port number to re-connect to, hence the wording of the binding update response in figure 10 "reconnect to port y". Once the new TCP connection has been established through the new interface, the session data transfer can continue.

3.2.7 Protocol State Diagram

Figure 11 shows the state diagram for the session layer protocol discussed above. The diagram shows the operation of the client (left-hand side, filled transitions) as well as the server (right-hand side, dashed transitions) at the session layer. State transitions can be triggered by three entities: the application program, a link-layer hint or network activity such as send or receive. The lower left-hand corner contains the logic for session redirection. In the diagram this is triggered by link-layer hint, but this does not have to be a pre-emptive hint – the message could be that the link-layer has gone down and so the new connection would be made through a different interface.

3.2.8 Multicast Support

Section 3.2.4 stated that the protocol does not support multicast. The reason for this is that one multicast IP address maps to many hosts. This one-to-many relationship means that a particular machine cannot be identified by the group IP and hence session redirection operations for a single machine cannot occur: session ids are only unique to a particular machine and therefore they cannot identify one particular machine on their own.

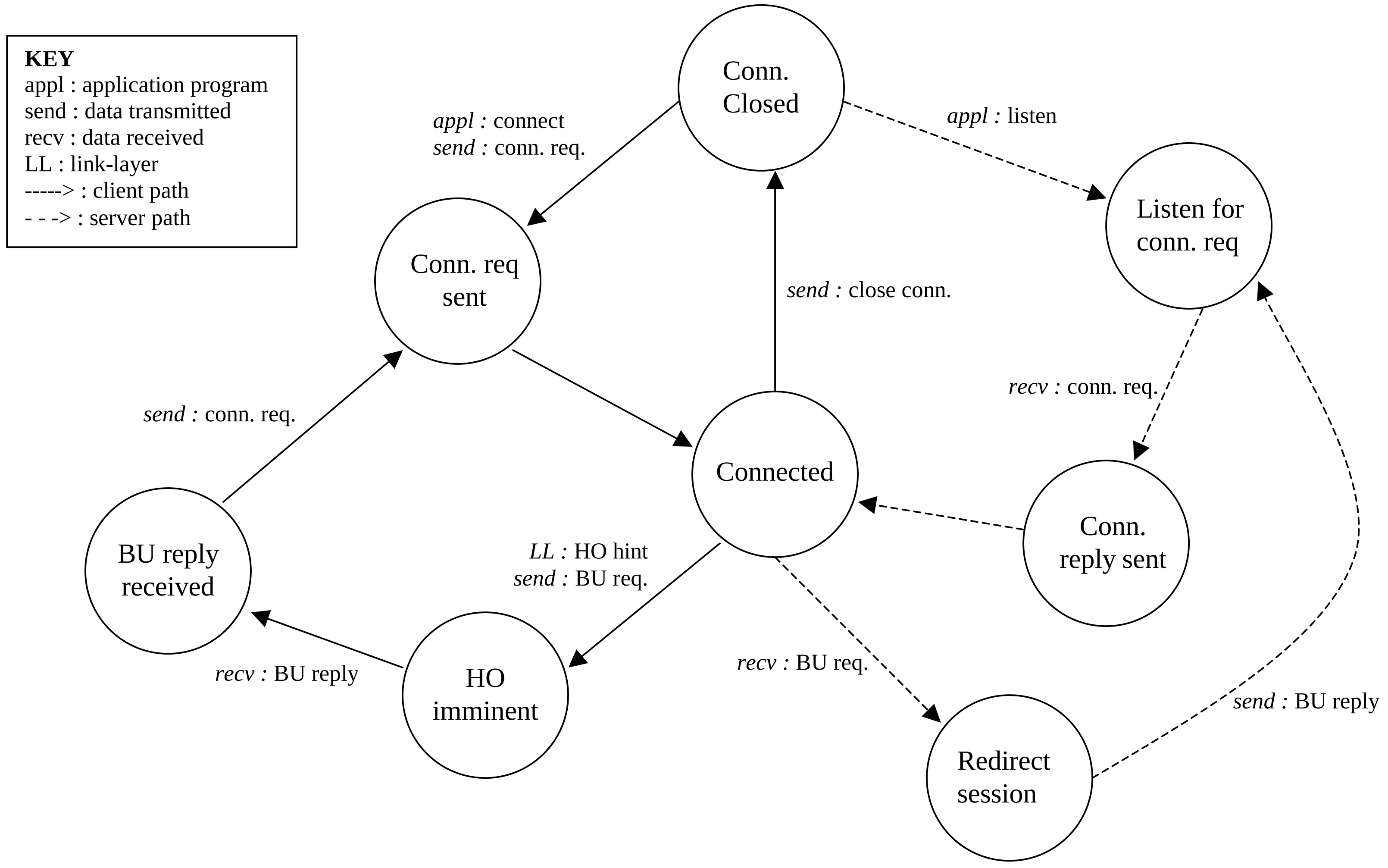

Figure 11 – Session Layer State Diagram

3.3 Protocol Implementation

The focus of this sub-section is to explain the issues concerning the implementation of the session layer protocol. Firstly, however, it is crucial to substantiate the necessity of implementing the protocol. The reason behind the implementation is to enable a performance analysis of the of the protocol in order to judge the design decisions made. There are many ways to analyse protocol performance. Computer simulation has the advantage that the scalability of the protocol can be easily predicted and results can be easily reproduced. However, simulation cannot portray the ‘real life’ workings of a protocol in the same way as an actual implementation can. Simulation cannot give a benchmark for the protocol running on a machine with distinct characteristics (processor, memory, network interface, services running) connected to a network with certain constraints and parameters (background load, latency, bandwidth and switching elements). Even more to the point, simulation cannot identify platform dependent ‘quirks’ such as a protocol not being able to cooperated with other protocols, services or hardware. An alternative method of performance analysis is mathematical modelling of the protocol operation. This method gives the theoretical bounds and limits of protocol performance, however, again it cannot take into account machine dependent limitations. As this thesis is a first endeavour into the design of the protocol it was decided that an implementation would give a good insight into the iterative process of protocol design. Simulation and mathematical modelling can be left as a future avenue to explore when the basis of the protocol has been created (see section 5.2).

3.3.1 Implementation Platform

The platform for the protocol implementation is the Intel PC architecture running Microsoft’s Windows 2000 Professional operating system. This platform was chosen due to the prevailing availability of personal computers and due to the ability of routing table modification within Windows 2000. Initially it was hoped to implement the protocol on the EPOC32 operating system running on a Psion 5mx handheld computer since this would be a more realistic mobile platform. However, after investigation of the EPOC32 architecture and the available network interfaces of the Psion 5mx it was decided that the platform, at the current time, is not suitable. Furthermore, due to the abstract nature of the Windows’ sockets architecture (below) it will be possible to port the protocol to future Windows Pocket PC (a.k.a. Windows CE) mobile devices.

3.3.2 Windows' Sockets Architectural Model and Interfaces

In order to support the multiplicity of available network subsystems an application requires a common, abstract interface with which it can transmit and receive data irrespective of the network stack. Sockets is the name for one Application Programming Interface (API) that provides this abstraction. Windows' Sockets (WinSock) is a non-proprietary, publicly available socket interface to the Microsoft Windows networking stack. The location of the WinSock API is shown in comparison to the OSI/RM in figure 12.

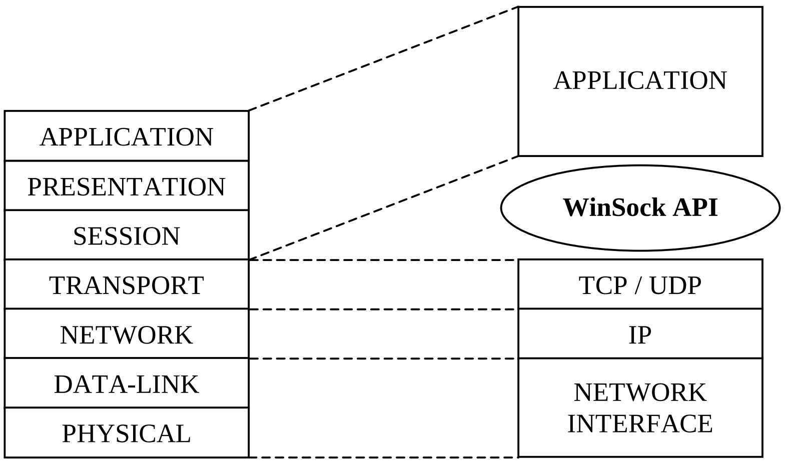

Figure 12 – WinSock API vs. OSI/RM

Although the transport and network protocols shown here are TCP / UDP with IP, the WinSock specification provides an interface to other protocols such as Asynchronous Transfer Mode (ATM), Internet Packet Exchange (IPX) / Sequence Packet Exchange (SPX), AppleTalk, Network Basic Input / Output System (NetBIOS) and IrDA.

The interface provided by WinSock is a set of functions that enable connection-less and connection-oriented client-server communication. The basic set of functions is given in table 1.

Function Name	**Description**
socket()	Creates a handle to the communications end-point (the socket)
bind()	Names a socket locally with the pair (IP, port number)
listen()	Listen for associations requests on a socket
accept()	Accept a connection request (connection-oriented sockets only)
connect()	Connect to a peer with address (IP, port number)
send() / recv()	Send / receive data buffer to / from an address (IP, port number)
sendTo() / recvFrom()	Send / receive datagrams to / from an address (IP, port number)
closeSocket()	Close the association

Table 1 – WinSock Functions

The functions are listed in approximate order of their use in a program. For instance before an application may listen() on a socket the socket handle must be created using socket() and the socket must be named using bind(). Further details of sockets programming can be found in much of the literature listed in the bibliography and an example of simple TCP and UDP senders and receivers can be found on the disk at the back of this thesis (see Appendix C - Contents of Accompanying Disk). The basic socket functions together with some more specialised ones are grouped into a dynamic link library (DLL) below the WinSock API [Hall97]. This API is used by every application written for WinSock, which constitutes all applications that use the Internet protocol suite.

Since the WinSock specification is compliant with the Windows Open Services Architecture (WOSA) it not only defines an interface for applications – the API – but also an interface for layered services – the Service Provider Interface (SPI).

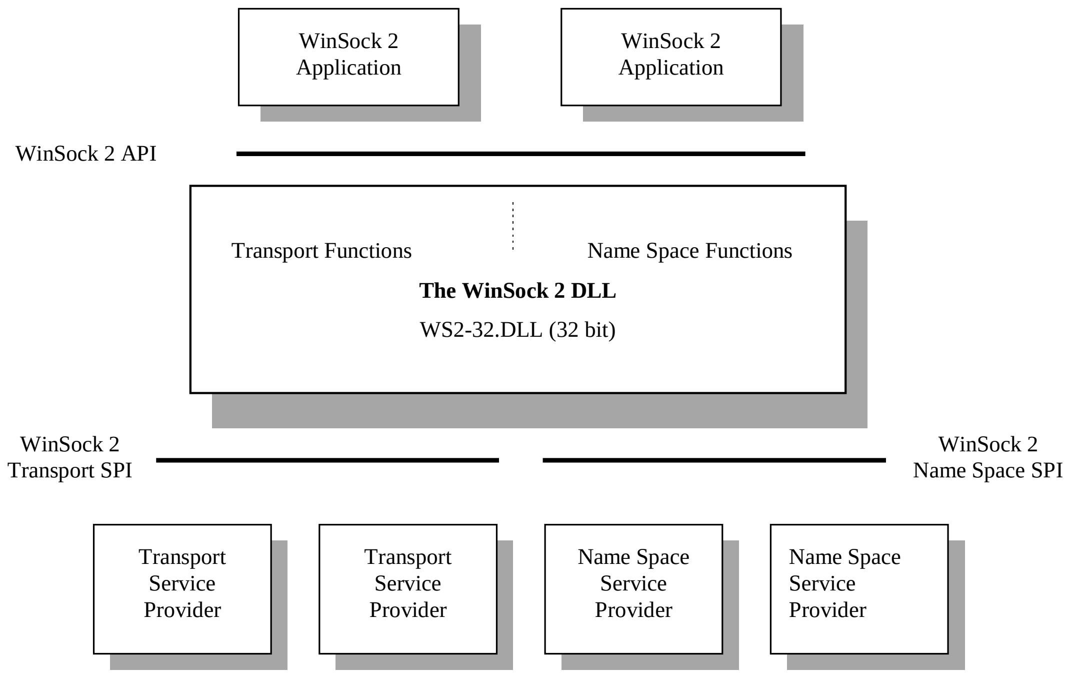

Figure 13 – The WinSock 2 WOSA-compliant Programming Interfaces (from [Hua99])

Figure 13 (from [Hua99]) shows how the WinSock API and SPI are WOSA-compliant. The Name Space SPI provides an interface to name service providers such as the Internet Domain Name System (DNS). The Transport SPI provides the interface to transport protocols such as TCP and UDP and is the SPI focussed on by this work

The architecture of the WinSock SPI allows transport providers to be layered above a base provider. The base provider is responsible for the implementation of the transport protocol workings: connection establishment and teardown, flow and error control and data transfer. Layered protocols – known as Layered Service Providers (LSPs) – can be chained above this base provider in order to supply added functionality to the base protocol. Internet protocols such as RSVP (for QoS resource reservation) and IPSec (IP Security) are implemented in this manner on the Windows 2000 operating system. The LSP model provides a convenient way of implementing the multi-homed management protocol since it allows the protocol to be layered above TCP and UDP whilst still using the WinSock API as a transparent interface into the protocol. Hence existing applications will operate transparently above the multi-homed management protocol without the need for modification. Figure 14 shows how the protocol integrates with the existing network stack as a LSP.

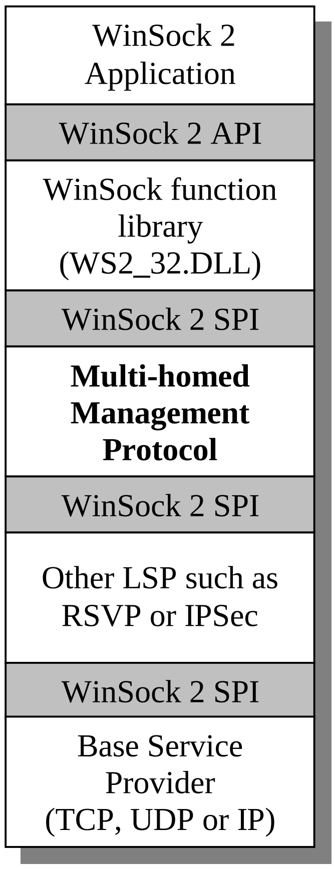
WinSock 2 Application
WinSock 2 API
WinSock function library (WS2_32.DLL)
WinSock 2 SPI
Multi-homed Management Protocol
WinSock 2 SPI
Other LSP such as RSVP or IPSec
WinSock 2 SPI
Base Service Provider (TCP, UDP or IP)

Figure 14 – Multi-homed Management Protocol in the WinSock Network Stack

Layered protocols are written to provide the WinSock 2 SPI to upper layers and to use the SPI provided by lower layers. In order to achieve this an LSP must provide a set of functions to match the SPI. This set of functions map onto the API functions (some of which are given in table 1 above). For each API function there is a corresponding SPI function (as shown in table 2).

API function	**SPI function**
socket()	WSPSocket()
bind()	WSPBind()
listen()	WSPListen()
accept()	WSPAccept()
connect()	WSPConnect()
send() / recv()	WSPSend() / WSPRecv()
sendTo() / recvFrom()	WSPSendTo() / WSPRecvFrom()
closeSocket()	WSPCloseSocket()

Table 2 – Mapping from WinSock API functions to SPI functions

As each API function is called the corresponding SPI function is called from the highest LSP in the chain. Each LSP in the chain calls the associated function provided by the next LSP down until the function in the base provider is called. Further facilities can be added to an LSP by modifying the set of functions that make up the interface. In this manner applications such as bandwidth managers and security modules can be created.

3.3.3 The Multi-homed Management Layered Service Provider

The protocol designed in section 3.2 has been implemented as a WinSock LSP. The two most important functions that have been added to the LSP are `GetAdapterToUse()` and `UpdateRoutingTable()`. The first function corresponds to the interface selection algorithm. It queries the operating system for a list of available network interfaces and according to the interface selection algorithm chooses the most appropriate one. To enable the algorithm to be changed it is implemented as a software component that can be modified without changing the function that queries the kernel. This function must be called every time `WSPSocket()` is called and to select a new interface during session redirection. The second function is used to bind a communications session to a specific interface. This is achieved by adding a static route into the routing table for the destination address of the session. Since multiple sessions to one destination may be bound to different interfaces the `UpdateRoutingTable()`function must be called each time a `WSPSend()` or `WSPSendTo()` function call is made.

The session PDU header is added and removed in the `WSPSend()` / `WSPSendTo()` and `WSPRecv()` / `WSPRecvFrom()` functions respectively. When session redirection is required the `GetAdapterToUse()` function is called to choose the new interface and the appropriate parameters are inserted into the PDU header, which is then transmitted to the peer.

A full copy of the commented program code can be found on the accompanying disk (see Appendix C - Contents of Accompanying Disk).

Chapter 4 - Protocol Performance

The protocol implementation described in the previous chapter has been evaluated empirically to determine its performance. To achieve this a network test-bed has been built and traces of the protocol operation have been captured.

4.1 Test-bed Architecture

A network test-bed was set up to observe the protocol in operation. Three subnets were created. Figure 15 shows the layout of the test-bed. Note that all IP addresses start with 143.167 but this prefix has been omitted from the diagram for simplicity. Each subnet had a server running Dynamic Host Configuration Protocol (DHCP) [Droms93] that assigned the mobile host an IP address dynamically. The numbers in boxes next to the DHCP server indicate the range of IP addresses assigned by that server. Each DHCP lease was set to an infinite amount of time so that DHCP address renewal messages would not interfere with the performance measurements. The subnets were addressed as show in table 3.

Subnet Number	**Network Address**	**Broadcast Address**	**Subnet Mask**	**DHCP Server Address**
1	143.167.11.184	143.167.11.191	255.255.255.248	143.167.11.186
2	143.167.11.168	143.167.11.175	255.255.255.248	143.167.11.170
3	143.167.11.176	143.167.11.183	255.255.255.248	143.167.11.178

Table 3 – Test-bed Addressing

A router was configured to exchange packets between the subnets and to provide name resolution capabilities (DNS). The router also acted as a network analyser to capture data-link layer frames from the Ethernet networks and decode them up to and including the UDP layer. Each end-system ran a WinSock monitor program that traces calls to WinSock API functions. The combination of the network analyser and the WinSock monitor were used to compute the timing information presented in the results. The computers used in the test-bed are shown in table 4 on page 45.

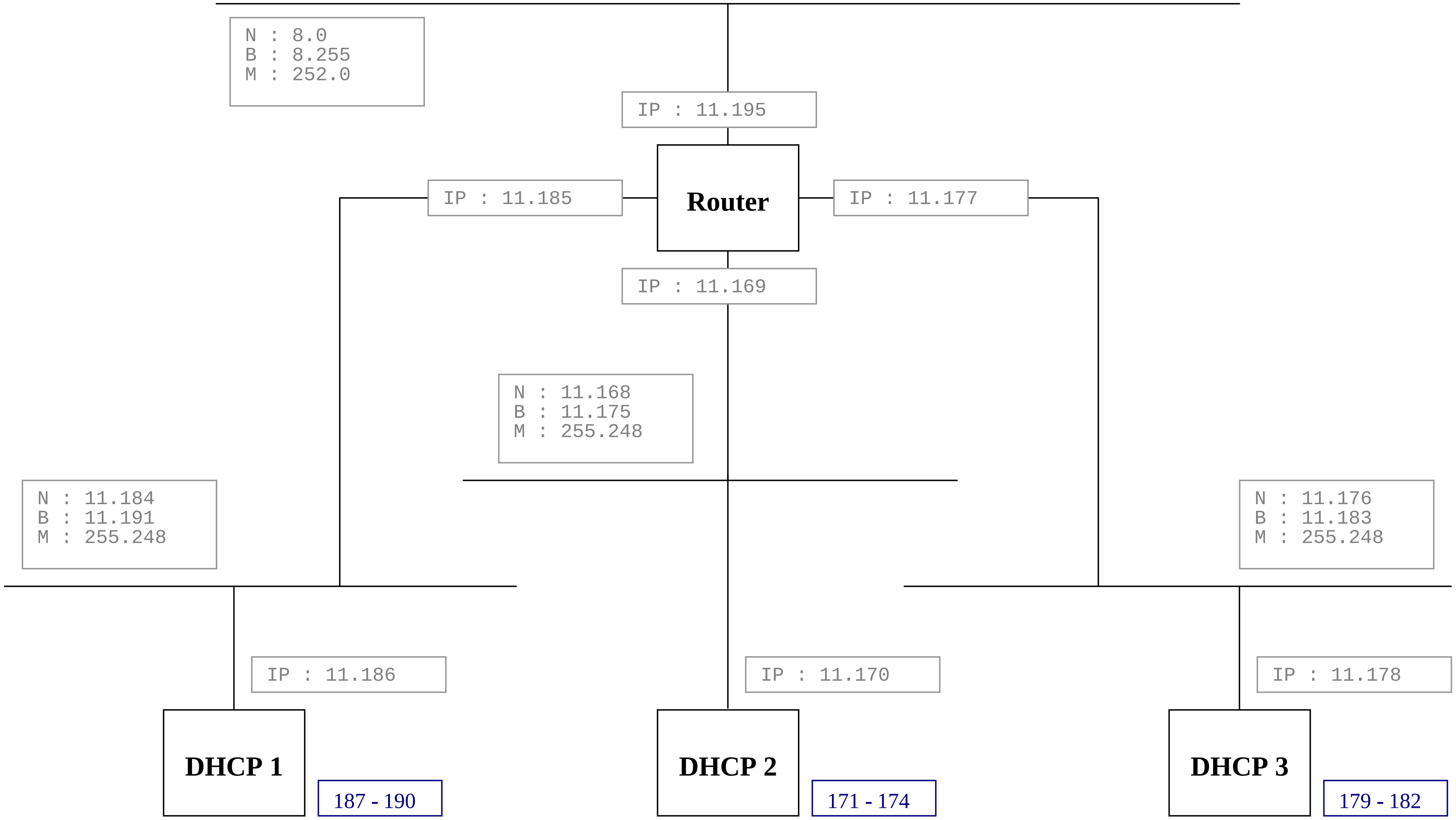

Figure 15 – Layout of the Test-bed

Machine	**Processor**	**Memory**	**Network Card(s)**	**Hard Disk**	**Operating System**
Client	Pentium II 233	256Mb	3Com (1 * PCI, 1 * ISA)	4Gb	Windows 2000 Professional
Server	Pentium 90	72Mb	3Com PCI	8Gb	Windows 2000 Advanced Server
Router	Pentium II 266	128Mb	3Com (2 * PCI, 2 * ISA)	3Gb	Windows 2000 Server

Table 4 – Test-bed Computers

Note that the DHCP servers are not shown in the table since they do not form part of the tests, they are only used initially to acquire an IP address. All machines were set up with a clean installation of the appropriate operating system before running any tests. The servers were configured with only the minimum services running – i.e. no web server, ftp server or mail server daemons were installed.

4.2 Description of Tests

The server was connected to the first subnet and the client to the second and third subnets. A simple UDP datagram program was written. The client program sent a datagram to the server, which echoed the datagram back to the client (see Appendix C - Contents of Accompanying Disk). The client program was used to send a series of ten datagrams to the server and the time taken for each one was calculated using the network analyser and the WinSock monitor. Measurements were taken with and without the protocol installed. The aim of this testing was to identify the delay caused on the machine by the extra functions contained in the session layer and to confirm that the actual frames transmitted on the network medium contained the correct session layer header information.

4.3 Analysis of Results

Table 5 and table 6 show the timing information as calculated from the network analyser. The datagram column shows the time (in milliseconds) from the echo to the previous datagram passing the analyser to the next datagram sent from the client passing the analyser. The echo column shows the reverse of this – the time difference between a datagram passing the analyser and the resulting echo being sent back. Hence the datagram column shows the time for the frame to be transmitted on the client subnet, the reception time at the client, subsequent processing by the client program and then transmission of the next datagram. Thus, the echo column shows the time for datagram transmission, processing and echo transmission on the server subnet.

	Datagram(ms)	**Echo(ms)**
Average	50.072	41.310
Standard Deviation	0.000	3.541

Table 5 – Network Analyser Timing (Normal Operation)

	Datagram(ms)	**Echo(ms)**
Average	190.274	257.036
Standard Deviation	0.001	20.846

Table 6 – Network Analyser Timing (Protocol Installed)

The results show that – on the client subnet – when the protocol is installed there is an increase of approximately 140ms in overall delay between frames. Since the client side subnetwork was isolated from any other traffic this delay is due to the added processing on the client during the `RecvFrom()` and `SendTo()` function calls. On the server subnetwork the increase is more pronounced: from 41ms to 257ms means that the delay has multiplied by just over a factor of six.

Table 7 and table 8 show the timing of the SendTo() and RecvFrom() functions under normal operation – i.e. without the protocol installed – as traced by the WinSock monitor. The trend from the results suggests that the SendTo() function takes 20ms whilst the RecvFrom() function takes half of this (around 10ms) yet with more variability (the standard deviation is between 3.16 and 5.65 ms).

	SendTo() (ms)	**RecvFrom() (ms)**
Average	20.000	9.111
Standard Deviation	0.000	5.645

Table 7 - WinSock Monitor Timing (Sender – Normal Operation)

	RecvFrom() (ms)	**SendTo() (ms)**
Average	10.000	20.000
Standard Deviation	3.162	0.000

Table 8 - WinSock Monitor Timing (Receiver – Normal Operation)

Table 9 and table 10 show the timing of the `SendTo()` and `RecvFrom()` functions (again traced by the WinSock monitor) with the protocol installed. The results show that the delay for the functions to complete has increased by an approximate factor of 10. However, there seems to be a disparity between the results measured by the network analyser and those measured by the WinSock monitor. The results from the network analyser show a delay of 190ms for a datagram echo reaching the analyser (at the router), being transmitted on the client subnet, being processed by the client and then the next datagram being transmitted on the client subnet and reaching the router again. However, the processing stage includes a call to both `RecvFrom()` and `SendTo()`, which would appear to take on average 377ms (211.333ms plus 165.5ms).

	SendTo() (ms)	**RecvFrom() (ms)**
Average	211.333	165.500
Standard Deviation	3.279	7.778

Table 9- WinSock Monitor Timing (Sender – Protocol Installed)

	RecvFrom() (ms)	**SendTo() (ms)**
Average	120.000	260.500
Standard Deviation	14.142	0.707

Table 10 - WinSock Monitor Timing (Receiver – Protocol Installed)

The reason for this is that the WinSock monitor only works at the API level and does not decode the actual SPI layer function calls. The added latency (compared with the network analyser trace) can be apportioned to the DLL function calls required to access the SPI layer. However, the results are still useful when compared to the control WinSock monitor timings (with the protocol not installed - table 7 and table 8). These results show that the latency due to the `WSPSendTo()` and `WSPRecvFrom()` functions in the SPI layer is an order of 10 greater than the control. The `WSPSendTo()` function must add the session layer header and call the `UpdateRoutingTable()` function. The `WSPRecvFrom()` function merely removes the header.

It was predicted, before the performance tests were carried out, that the implementation would add latency to data transfer, and this latency would be due to the `UpdateRoutingTable()` function. However, the results seem to show that the latency of this function is minimal since both `SendTo()` and `RecvFrom()` suffer from an increased delay. It is thought that the operation of the SPI layer as a whole causes the added latency in operation. This raises the question as to whether the WinSock SPI layer is a good way of implementing layered protocols on top of TCP or UDP. Perhaps a more efficient way would be to alter the kernel-level code, but this raises difficulties in itself due to compatibility with existing systems since changes to the kernel cannot be installed as easily as the SPI layer DLL.

Chapter 5 - Conclusions

5.1 Summary of results

The results show that the latency of data-transfer is increased by a factor of ten due to the protocol operation. It is believed that this dramatic increase is due, not only to the protocol functions, but also to the operation of the WinSock SPI layer. It is believed that use of the SPI layer is not a good method for implementing layered protocols on top of TCP/IP and that a better way may be to actually modify the kernel layer device drivers to incorporated the protocol. The work of Zhoa [Zhoa98] discussed in section 2.5.1 made modifications to the Linux kernel routing table to accomplish multi-homing and this may be the way to go in the future.

5.2 Future work

The following areas are open for future investigation following this project.

- Link-layer Hint: the mechanism for monitoring the power level of the received BTS signal could be investigated to see if it is possible to provide such a hint. If this has potential then the timeliness of the hint could be researched so that session redirection can be accomplished before the link-layer goes down.
- Security of binding update packet: some method of authenticating the mobile is needed to enable binding updates to be used securely. If not, the protocol is vulnerable to counterfeit binding update messages, which could enable a user to eavesdrop any data transmitted from the mobile.
- Interface selection algorithm: section 2.6 described the integration of QoS, security and billing within a multi-homed architecture in which the choice of interface is made according to the traffic characteristics and application demands. The user should be able to select values for the relative importance of QoS, security and billing on a per-flow basis, which the interface selection algorithm should take into consideration.
- Simulation / mathematical modelling: *what is the scalability of the protocol? Will it still work with many users in a cell at a given time?* The implementation can only test the protocol with a limited number of mobile

hosts connected to a specific medium. However, with a computer simulation or mathematical model the number of active users, number network interfaces per mobile and cellular network architecture can be varied.

5.3 Project Evaluation

It is felt that the project has been a success. However, with any software engineering project time is the key factor. Regrettably, the process of writing the implementation of the protocol took longer than first expected and as such the stage of testing the protocol and analysing its performance suffered. Although the performance of session redirection could not be evaluated due to time constraints and problems with DHCP address release, the performance of the implementation has been considered and shown to be heavily dependant on the SPI architecture. In the context of this work this is a negative point, however, it has been important to discover these problems so that they can be avoided in future work.

On a personal level the project has formed a good education in mobile communications. Learning about C++, sockets programming and different data-link level technologies in a year has been difficult but rewarding.

Appendix A - References

WAPForum99	WAP Forum, "WAP White Paper", 1999
ETSI99	European Telecommunications Standards Institute, *Digital cellular telecommunications system (Phase 2+); General Packet Radio Service (GPRS); GPRS Tunnelling Protocol (GTP) across the Gn and Gp Interface; (GSM 09.60 version 6.4.1 Release 1997)*
Nokia99	Nokia Corporation, *IP-Radio Access Network - Nokia's vision for an all IP based architecture for Radio Access Networks*. Nokia White Paper, 1999.
Perkins96	Perkins, C.E. (Editor), et al., *IP Mobility Support*, Internet Engineering Task Force Request For Comments Number 2002, 1996.
UMTSForum99	UMTS Forum, *Universal Mobile Telecommunications System*, 1999.
Bluetooth99	Bluetooth Consortium, *Specification of the Bluetooth System Version 1.0 A*, 1999.
Scourias97	Scourias, J., *Overview of the Global System for Mobile Communications*. University of Waterloo, Department of Computer Science, August 1997.
Droms93	Droms, R., *Dynamic Host Configuration Protocol*. RFC 2131, 1993.
Simpson93	Simpson, W. A., *The Point-to-Point Protocol (PPP).* RFC 1548, December 1993.
Romkey98	Romkey, J. L., *A Nonstandard for Transmission of IP Datagrams Over Serial Lines: SLIP*. RFC 1055, July 1988.
Hamalainen96	Hamalainen, J., PhD. Thesis, University of Tampere, Finland, 1996.
Huitema97	Huitema, C., *IP Version 6: The New Internet Protocol*. Prentice Hall, 1997.

Postel81a	Postel, J. B., *Internet Protocol*. RFC 791, September 1981.
Postel81b	Postel, J. B., *Internet Control Message Protocol*. RFC 792, September 1981.
Postel84	Postel, J., "Multi-LAN Address Resolution", RFC 925, October 1984.
Zhoa98	Zhoa, X., Castelluaccia, C. and Baker, M., *Flexible Network Support for Mobility*. Proc. of Mobicom, (Dallas, Texas), Oct. 1998.
Katz96	Katz, R. H. and Brewer, E. A. *The Case for Wireless Overlay Networks*. Proc. 1996 SPIE Conference on Multimedia and Networking, MMCM '96, San Jose, CA, (January 1996).
IEEE99	Institute of Electrical and Elecronics Engineers, *Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 1999.
ETSI98	European Telecommunications Standards Institute, *Broadband Radio Access Networks (BRAN) HIgh PErformance Radio Local Area Network (HIPERLAN) Type 1; Functional specification* Doc. Nb. EN 300 652 Ver. 1.2.1 Ref. REN/BRAN-10-01, 1998.
IrDA96	Infrared Data Association, *IRDA Serial Infrared Link Access Protocol (IrLAP) Specification*. Version 1.1, June 16, 1996
Stemm98	Stemm, M. and Katz, R.H., *Vertical Handoffs in Wireless Overlay Networks*. ACM Mobile Networking (MONET), 1998.
Maltz98	Maltz, D.A. and Bhagwat, P., *MSOCKS: An Architecture for Transport Layer Mobility*. Proceedings of the Conference on Computer Communications (IEEE Infocom), (San Francisco, California), pp. 1037, March/April 1998.
Pahlavan99	Pahlavan, K., Krishnamurthy, P., Hatami, A., Ylianttila, M., Mäkelä, J., Pichna, R. and Vallström, J., *Handoff in Hybrid Mobile*

	Data Networks. To appear in IEEE Personal Communications Magazine.
Inouye97	Inouye, J., Binkley, J. and Walpole, J., *Dynamic Network Reconfiguration Support for Mobile Computers*. Mobicom, ACM, ed., 1997.
Stewart00	Stewart, R. R. et al., *Stream Control Transmission Protocol*. IETF Internet Draft, July 2000. Expires December 2000.
ISO84	International Standards Organisation, ISO 7498-1, *OSI Basic Reference Model*. 1984.
Halsall96	Halsall, F., 1996, *Data Communications, Computer Networks and Open Systems*. Addison-Wesley, pp. 697 – 706.
Postel80	Postel, J. B. (editor), *User Datagram Protocol (UDP)*. RFC 768, August 1980.
Postel81c	Postel, J.B. (editor), *Transmission Control Protocol (TCP)*. RFC 793, September 1981.
Hall97	Hall, M. (Editor), *WinSock 2 Application Programming Interface Specification*. Stardust Technologies, Revision 2.2.2, August 7 1997.
Hua99	Hua, W. et al., *Unravelling the Mysteries of Writing a Winsock 2 Layered Service Provider*. Microsoft Systems Journal, May 1999.

Appendix B - Bibliography

- Jones, A. and Ohlund, J., 1999, *Network Programming for Microsoft WinSock*. Microsoft Press.
- Qunin, B. and Shute, D., 1996, *Windows Sockets Network Programming*. Addison-Wesley.
- Stevens, W.R., 1994, *TCP/IP Illustrated Volume 1 : The Protocols*. Addison-Wesley.
- Stevens, W.R. and Wright, G.R., 1995, *TCP/IP Illustrated Volume 2 : The Implementation*. Addison-Wesley.
- Stevens, W.R., 1990, *Unix Network Programming*. Prentice Hall.
- Eckel, B., 2000, *Thinking in C++*. Prentice Hall.
- Daconta, M.C., 1995, *C++ Pointers and Dynamic Memory Management*. Wiley-QED.
- Perkins, C., 1998, *Mobile IP Design Principles and Practises*. Addison-Wesley.
- O’Grady, V. (Editor), 1999, *GSM World Focus*. Mobile Communications International.

Appendix C - Contents of Accompanying Disk

An MS-DOS formatted floppy disk is contained in an envelope fixed into the back of this thesis. The disk contains three things:

1. An Adobe PDF file containing the text of this document
2. A copy of all program code written during the project
3. A copy of the raw results captured from the network analyser

The second item is arranged as follows:

- ❑ The directory ‘LSP’ contains two subdirectories: ‘LSP’ and ‘instLSP’. The former is the code for the actual layered service provider and the latter is an installation program for the LSP.
- ❑ The ‘dataTest’ directory contains the programs used to test the protocol. These are simple WinSock API programs that merely send and receive data depending on command-line options.
- ❑ The ‘Install’ directory contains all of the files needed to install the protocol on Windows 2000.

To install the protocol copy the contents of the Install directory to the `WINNT/System32` directory on the computer; then run the application “instlsp -install”. The layered service provider can be removed by running the program “instlsp -remove”.

Appendix D - Original Project Proposal

The original proposal for the MSc. (Research) project is attached following this page. The project has changed completely since this proposal was written due to a change in academic supervisor just before the commencement of the project. As a result the following report is for reference purposes only.