Network Support for Mobile Hosts in a TCP/IP Internetwork

A Thesis Submitted to the College of

Graduate Studies and Research

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in the

Department of Computer Science

University of Saskatchewan

Saskatoon, Saskatchewan

by

Vineet Chikarmane

August 1995

PERMISSION TO USE

Network Support for Mobile Hosts in a TCP/IP Internetwork

Candidate: Vineet Chikarmane

Supervisors: Richard B. Bunt and Carey L. Williamson

Master of Science Thesis

Presented to the College of Graduate Studies and Research

University of Saskatchewan

August 1995

ABSTRACT

There is an increasing need to support user mobility in today's computing environments. Packet-switched computer networks are commonplace and are used to provide services to users who, increasingly, may be mobile. Hence, routing packets to mobile hosts is a problem of considerable research and commercial interest. Of significant interest are protocols like those intended for TCP/IP networks in general and the Internet in particular. This thesis deals with approaches to supporting mobility in a TCP/IP internetwork.

A split-level design is presented for providing routing support for mobile hosts. A prototype implementation of a unicast mobile routing protocol was implemented in a 4.3 BSD Unix™ operating system. The performance of the mobile routing subsystem was evaluated using measurements by suitably instrumenting the kernel and using a framework developed for performing the experiments. The mobile routing subsystem was found to perform well in the context of common network technologies such as Ethernet.

The basic unicast routing capability can also serve as the foundation for the design of a multicast service facility for mobile hosts. The scheme presented leverages existing IP multicast for static hosts to provide multicast services for mobile hosts as well. The scheme supports dynamic groups and preserves IP multicast semantics.

ACKNOWLEDGEMENTS

I would like to thank my supervisors, Dr. Rick Bunt and Dr. Carey Williamson for their technical guidance and encouragement throughout my graduate studies. Without them, this thesis would not have been possible. The other members of my thesis committee, Dr. Derek Eager and Dr. Janelle Harms provided insightful comments on my work. I am grateful to *TR*Labs for providing me with excellent facilities and financial support, and to the Center for Advanced Studies at IBM Canada for their generous financial support.

My friends helped with lots of things, and grad life would have been pretty tough without them. Prithvi got me started with Unix kernel hacking. Tim Harrison patiently put up with my constantly changing equipment configuration requirements. Venkat Josyula and Arun R. helped with the final thesis draft and post-processing while I was away. Dwight Makaroff helped me put my thesis defense presentation together. Kevin Froese generously agreed to do the binding of the final thesis document.

My family was a constant source of support and advice. My mother Meera made my Masters and most things in my life possible. My brothers Sanjay and Vinay, and my sister Supriya were always full of practical suggestions. Finally, Deepa came all the way around the world to check out on me. I could always count on her to be at the far end of the international phone-line whenever I needed her. She made me and SaskTel very happy indeed.

I dedicate this thesis to my parents.

Contents

List of Figures

Chapter 1

Introduction

Over the years, connectivity to global resources through computer networks has become commonplace. Indeed, it has become a necessity for many users, providing them with services such as electronic mail and file transfer. In recent years, general information services provided through applications like the World Wide Web [BLCGP92] have seen dramatic increases in popularity and use. Computer networks are increasingly being used as a vehicle for the dissemination and collection of information of general interest to and from a huge number of people.

Today's user community is demanding a level of mobility not previously anticipated by designers of computer networks. Simultaneously, hardware technological advances have made inexpensive and powerful portable computers and wireless networks [IB94] a reality. This convergence of a need for mobility and products to satisfy that need has created an explosion of interest in problems related to the mobile computing paradigm [FZ94].

A *portable computing device* is one that has features that make it easy, practical and useful to move it to different locations. A *mobile user* is one who requires internetwork access at different places at different times. A *mobile computing device* is one that is capable of providing services in the face of user mobility, even when movement is actually taking place. The gamut of issues that must be dealt with in the provision of these services is referred to as *mobility management*. Given that network access is crucial, a necessary feature of mobility management is the continuous provision of network services. The provision of effective network support for mobility will provide the basis for the creation of new classes of applications and, potentially, new classes of users. Wireless networks are widely expected to increase the transparency of network access and make network access ubiquitously available to users who may be mobile. However, wireless networks do not necessarily imply mobility and mobility does not necessarily imply the use of wireless networks.

As a mobile user wanders around an internetwork and attaches to it at different locations at different times, it is properly the responsibility of the network software to provide mobility support in a transparent and seamless fashion. In particular, the following issues are of research and commercial interest:

- **Communications Management issues**: These are concerned primarily with the implications of the different network access mechanisms that may be employed. Here, issues such as bandwidth management and channel access in the context of wireless communication are of interest. While mobility subsumes both wired and wireless mobility, wired networking has been well researched and many of the issues are well understood. Wireless networking poses new challenges in communications management that have to be addressed.

- **Location Management issues**: Issues arising from the need to deal with the changing *location* of the user are referred to as *location management*. Location management will become even more important as wireless network interfaces become widely available since, once the computing devices are unconstrained by cable, it is likely that frequent user movement across networks will become common. Host "location" may not even be under the user's control. For example, if a mobile host operates in overlapping wireless cells, the host may move from one cell to another automatically, based on dynamic factors such as load and noise.

- **Service issues**: Mobile users need to be able to access services at different locations in a transparent manner. The ability of users to *roam*, i.e., to move from one network to another without advance notice, causes potential breaks in services. New service architectures are needed to provide effective services in the face of user mobility.

1.1 Background

This section identifies the various issues pertinent to this research.

The seven layer *International Standards Organisation* (ISO) *Open Systems Interconnection* (OSI), network reference model [DZ83] is used to conceptually layer the functionality of computer networks. The seven layers are: the Application layer, the Presentation layer, the Session layer, the Transport layer, the Network layer, the Data-link layer and the Physical layer. In two communicating machines, a layer on one machine may be considered to communicate with the corresponding layer of the other machine using the *protocol* of that layer. A collection of layered protocols constitutes a *protocol stack*. Today's networking protocol stacks, including the *Transmission Control Protocol/Internet Protocol* TCP/IP [Tan88] and the OSI [Neu83] protocol suites, generally assume that each computer attaches to the network at a fixed physical location.

Networks may be interconnected to form internetworks, the most widespread being the Internet which uses the TCP/IP protocol stack. In terms of the ISO OSI reference model, TCP is a Transport layer protocol and IP is a Network layer protocol, as shown in Figure 1.1.

The interconnection of networks (the internetworking function) is achieved in the Network layer of the ISO OSI reference model, corresponding to IP in the TCP/IP network stack. The Network layer protocol is responsible for delivering information across the internetwork in discrete chunks called *datagrams*.

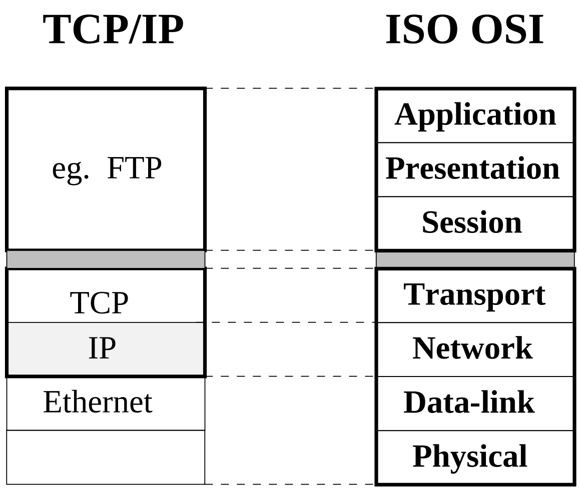

Figure 1.1: TCP/IP in the context of the ISO OSI model

The location of a computer on an internetwork is associated with a Network layer *address*. This address is also used for uniquely identifying a host's network interface on an internetwork, and thus encodes both location information as well as identity in one entity. Each datagram is individually *routed* through the internetwork using the Network layer address of the destination, in a manner analogous to a letter in the postal system being delivered to a recipient using the address of the receiver. Since identity and location are fused into a single address, host movement causes misrouting. In protocols today, host movement is assumed to occur so rarely that it can be handled manually, by suitably reflecting the new location of the host in the Network layer address of the host.

Communication between nodes across an internetwork can take several forms. *Unicast* is defined as the sending of a datagram from a source to one destination i.e., *one-to-one* communication. When a source must send a datagram to many destinations, repeated transmissions of the same datagram to each of the multiple destinations wastes resources such as network bandwidth and host processing cycles. These resources may be scarce, especially in a mobile computing environment in which wireless links have limited bandwidth and portable computers have limited resources. *Multicast* is a mechanism for *one-to-many* communication in which the source transmits a single datagram, and the network performs the task of delivering that datagram to the multiple destinations. It is a technique that can potentially save both network bandwidth and host processor cycles.

Issues relating to unicast to mobile hosts have been of considerable interest to the networking research community, and several proposals for handling mobile unicast have been presented. The proposals differ in the manner in which location information is represented and propagated, and the details of the mechanisms used to forward packets to mobile hosts. Myles and Skellern [MS93] evaluated three proposals [IDM91, PR92, TCT92] based on compatibility with existing infrastructure, route optimality, migration procedures and other considerations. Subsequent work [MP93, Joh94, JMP94] sought to address the problems of the earlier proposals in areas such as robustness, scalability and security.

1.2 Goals of this Research

An Internet Engineering Task Force (IETF) Working Group has been chartered to coordinate efforts towards the specification of a protocol, **Mobile IP** [Per95], to handle host mobility in the Internet. This protocol is still in the process of being specified but is being developed as an Internet standard. As of the middle of May 1995, no implementations of Mobile IP existed in the public domain. Thus, one of the main goals of this thesis work is a design specification and a prototype implementation providing routing support using Mobile IP.

Very little has been done in the area of multicast for mobile hosts and only one earlier proposal [AB93] exists. It was a goal of this research to specify and evaluate a new proposal to provide multicast [CBW95], itself a Network layer function, as a service to mobile hosts. The new proposal is based on a foundation of the Mobile IP unicast mobile routing protocol.

This research is focussed exclusively on mobility in the context of the TCP/IP protocol stack since it is the predominant protocol stack in use. Furthermore, source code implementations of the TCP/IP protocol stack and published literature about it are widely available making it the ideal vehicle for research. Ioannidis [Ioa93] identified location management as being primarily a Network layer problem since host movement introduces problems in the routing of packets, which must be addressed in the Network layer. This research investigates issues in providing **Network layer support** (indicated by the shaded region in the TCP/IP stack in Figure 1.1), including both unicast and multicast routing, for mobile computing devices in a TCP/IP internetwork. In particular, this work focuses on IP as the Network layer protocol, and implicitly assumes the existence of a Data-link layer protocol such as Ethernet below IP.

The goals of this thesis are:

1. A survey of proposals for handling host mobility in a TCP/IP network, one of which (Mobile IP) was selected as a basis for implementation.
2. The design of network layer modules to provide unicast routing support for mobile hosts.
3. The implementation of the routing support in the context of a 4.3 BSD based Unix kernel.
4. Measurements to determine the performance of the implementation.
5. The definition of requirements of multicast support for mobile hosts.
6. The design of a scheme for providing multicast as an example service using Mobile IP as a foundation.

1.3 Organisation of the Thesis

In the next chapter, a general introduction to mobile computing is presented and some of the problems introduced by mobility are described. In Chapter 3, different approaches to supporting mobility at the network layer are surveyed, and an overview of the specific one that this research work is concerned with, Mobile IP [Per95], is given. In Chapter 4, a high-level architecture of the system is presented and the design of a routing subsystem to provide unicast routing support for mobile hosts is described. In Chapter 5, the protocol interactions required to enable mobile routing are examined and their impact on the design are considered. In Chapter 6, the implemented router is evaluated using controlled tests, and measurement experiments are conducted to evaluate the performance of the router. In Chapter 7, the basic Mobile IP routing capability is used as a foundation to propose a solution for multicast for mobile hosts [CBW95]. Finally, Chapter 8 presents a summary of the work done in this thesis, the conclusions drawn from it, and the contributions made by it. Future research directions are also suggested. Two Appendices are provided to present an algorithmic view of the router design and measurement setup. Finally, a glossary of the important terms and acronyms used in this thesis is also provided.

Chapter 2

Mobile Computing

Mobile users will use different kinds of mobile hosts or devices, and will run different kinds of applications to achieve different goals. Several classes of end-user mobile hosts and applications exist. In this chapter, some products, services and applications relevant to mobile users are reviewed.

2.1 Mobile Computing Devices

Two major categories of mobile computing devices or mobile hosts are of interest:

- **Portable Computers**: These include laptops, palmtops and personal digital assistants (PDAs). These are general purpose computing platforms used for convenience and their inherent suitability for work involving more than one location. They are becoming increasingly powerful in terms of processing power and storage. With improvements in screen technology, these machines are emerging as attractive alternatives to desk-bound computers. For these devices, network access is a natural extension of the raw computing environment they provide to users.

- **Embedded Computers**: These include controllers and processors that are integrated into machinery, cars, trucks and military vehicles. They have very specific applications, and network access is used primarily for control and monitoring functions.

2.2 Mobile Computing Applications

Several possibilities for mobile applications exist. Traditionally, mobile data communications applications have been written for specific domains such as truck and vessel fleet dispatch and management. Increasingly, independent applications for mass use are appearing that can be categorised as *information service applications* or *mail-enabled applications*. Some important applications of mobile computing are summarised briefly in the following subsections.

2.2.1 Enterprise Management Information Systems

Many corporate restructuring efforts have occurred over the last few years, aimed at making organisations leaner, more productive and faster in responding to market needs. Several mobile computing applications are directly applicable in this scenario. The goal is to profit from reduced expenditures on office real-estate and infrastructure while simultaneously enabling timely information exchange between personnel in the field and corporate divisions so as to provide a competitive level of service.

Sales information networking Sales representatives can be informed of customer needs when on the move and can in turn keep in touch with the office to obtain information from central databases regarding customer orders or product availability. In an integrated network, orders can be verified and processed immediately, monetary transactions effected immediately, inventory cross-checked, raw materials ordered, production scheduled and products dispatched.

Service information networking Service and maintenance personnel can be alerted with respect to specific customer requirements, and can in turn access company-wide resources to perform tasks such as solving technical problems and ordering replacement components.

2.2.2 Multi-location Work

Several lines of work such as consulting or academe/research often entail frequent travel between multiple locations for technical discussions, conferences, and the like. Increasingly common extensions of this scenario are the notions of work-at-home, the home-office and teleworking. However, users are faced with the prospect of working in different environments when computing facilities exist at more than one location.

Portable computers offer the opportunity to maintain similar environments by virtue of using the same machine at multiple locations. *Disconnected operation* [KS92, HH93, HPGP92] makes use of caching techniques to provide access to some file system functions normally available through a distributed file system, even when the user's computer is not physically connected to the network. This technology effectively provides the same distributed file system environment for users of portable computers, regardless of location, and constant access to the file server is no longer needed. However, disconnected operation has its limitations [Chi94], including exhaustion of resources on the disconnected client and the eventual staling of the cached files. More importantly, disconnected operation implies that a critical component of a user's environment, i.e., the ability to communicate through a network, is absent. A network connection, where available, circumvents all these problems, enabling both communication and file system functions like remote file down-loading, server reconciliation and uploading of files that cannot be accommodated locally. Finally, it makes possible access to other locally available network resources like shared printers.

2.2.3 Personal Services

Several classes of information are of relevance when presented to the user in a timely or location-specific manner. These include, for instance, the status of personal stock and investment portfolios; weather information and forecasts; for commuters, traffic conditions, congestion avoidance information and dynamic car-pool organisation; for tourists, travel maps and information on locally available facilities and services; for travellers, flight, bus and train schedules; for shoppers and consumers, product and local promotions information. As the user moves from one location to another, the network must allow the mobile computing device to be correctly addressed and ensure that messages from the relevant information servers are correctly delivered to the mobile user.

These services will be provided by advanced agent-based networks [EW94], where software in the network (*agents*) provides information to and from the mobile computing devices based on user-supplied *profiles*. Agents take care of issues such as

- *inter-operation* with different classes of mobile computing devices, such as PDAs, laptops, pagers with possibly differing network access and interface technologies, and differing user interfaces.
- *filtering* to provide users only the information in which they are interested. For example, an investor may want to be informed only of sudden drops in the stock that the investor owns, so that immediate action can be taken.
- *buffering* in the event that the target mobile device is temporarily inaccessible.

These advanced agent-based networks will provide a flexible architecture on which specific applications can be engineered. Companies such as IBM and AT&T are in the process of deploying first-generation networks based on this technology.

2.2.4 Other Services

The Walkstation project [RIOP94] at BBN (with ARPA sponsorship) involves the development of a system with applications to military/police team coordination and medical monitoring. The missions addressed by the military/police teams include counter-terrorism actions, drug inter-diction, and hostage-rescue efforts. The medical monitoring aspect involves plotting a person's geographical position and basic health status on an electronic map using a wireless packet network. By having both medical information and position information, people using the system can send help to others who are injured or incapacitated as soon as their medical status falls out of a predetermined range, thereby speeding triage and treatment. This has applications in military units, in emergency services organisations and in health care. Medical monitoring can also be used to present information about patients to doctors who are typically mobile, alerting them to specific medical conditions of the patients under their supervision. While proceeding to, say, an operating room, the doctor can schedule emergency procedures to be undertaken by other personnel prior to the doctor's arrival.

Embedded computers are already being used to perform mechanical checks on components of trucks and communicating the information to service points ahead on the route being taken so that replacement or other procedures can be scheduled and undertaken, saving money owing to total breakdowns and delays, and increasing the reliability of the transportation service.

At least one major courier company is using mobile computing to track individual consignments as they make their way to the destination, making it possible to provide precise delivery estimates, reliability due to elimination of misrouting, and a superior level of customer service. Similar tracking systems have applications in theft control for expensive products like cars.

2.3 The Routing Problem in Mobile Computing

This thesis deals primarily with routing for mobile hosts. This section provides the context for the research work by presenting a brief description of unicast and multicast routing in today's TCP/IP networks. The impact of mobility on these existing mechanisms is then described.

2.3.1 Unicast Routing

Datagram Routing In a TCP/IP internetwork, individual networks are interconnected using IP *routers*. In order to make correct forwarding decisions for datagrams, routers exchange information with each other. *Routing protocols* such as RIP [Mal94] and OSPF [Moy94b] serve to disseminate routing information across an internetwork. Based on this information, routers make routing decisions for individual datagrams. As shown in Figure 2.1, IP addresses consist of a network component and a host component.

Figure 2.1: Network component of an IP destination address in an IP datagram

The two components serve to uniquely identify a specific network in an internetwork and a specific host on that network. In order to bound the sizes of routing tables at the routers and the volume of traffic generated by the routing protocols, datagrams are routed using only the network components of IP addresses. Every host and router must make a *routing decision* for all outgoing packets, which basically determines the unique network interface that must be used to transmit the outgoing packet. If the source and destination are attached to the same physical network, then **direct routing** is said to occur since the datagram can be received directly by the destination. If the source and destination of a datagram are on different networks, then the routing decision basically determines the correct intermediate router (the "next hop" router) on the route to the final destination. In this case, **indirect routing** is said to occur and the datagram is passed to the intermediate router for further routing. However, in both cases, the datagram must be directly delivered to the receiver across a single physical network.

Sometimes, datagrams may also need to be "delivered" indirectly when the sender and receiver are separated by one or more intermediate routers. This kind of delivery can be used in situations such as those where the intermediate routers would make incorrect routing decisions based on information that would normally be correct, but is incorrect due extraordinary circumstances. Host mobility is one such scenario where this kind of delivery can be effectively used.

Datagram Delivery On shared media LANs like Ethernet, interface hardware on computers performs filtering based on the MAC (Medium Access Control sublayer of the Data Link layer in the ISO OSI Reference model) address in the destination fields of the frames as seen on the LAN. Hence **direct delivery** of the IP datagram is achieved by encapsulating it in a frame with the frame destination address being the MAC address of the receiver. In TCP/IP, the IP addresses of receivers are mapped onto corresponding MAC addresses using the Address Resolution Protocol, ARP [Plu82].

Tunneling is a mechanism by which packets can be delivered across a complex internetwork. Tunneling of a high level protocol such as IP itself can be achieved by using either *source routing* or *encapsulation*. Source routing achieves delivery by explicitly specifying the path to the destination of the datagram in the header of the datagram, using a list of intermediate routers. Encapsulation prepends the entire datagram to be delivered with another header, thus treating the data as well as the original header as data. Tunneling thus serves as an **indirect delivery** mechanism. Intermediate routers (other than the routers specified in the source route, if source routing is used) inside the tunnel are not aware of the existence of the tunnel. A tunnel functions as a virtual point-to-point link between tunnel end-points that are not on the same physical network. Datagrams may be considered to be delivered from one tunnel end-point to the other across this virtual link using a software interface. The set of hardware interfaces and software interfaces that a machine maintains may be treated as *virtual interfaces*, a term commonly used in protocols where tunneling is employed.

2.3.2 Multicast Routing

Deering [DC90] designed a form of WAN multicast that has been incorporated into the TCP/IP suite. Conventions have been defined for recognising a particular class of IP addresses, the Class D IP addresses, as being *multicast addresses* and for mapping these addresses to MAC layer multicast addresses. The Internet Group Management Protocol, IGMP [Dee89], is used by hosts to inform multicast routers about the multicast addresses that the hosts wish to receive. A group of hosts that might wish to communicate with each other through a given multicast address is called a *host group*. A multicast routing protocol, usually in the form of an extension to an existing routing protocol, is executed on routers to construct datagram delivery paths and to accomplish datagram forwarding. DVMRP [WPD88], a distance-vector routing protocol, has been implemented as an extension to RIP, and MOSPF [Moy94a], a link-state routing protocol, has been implemented as an extension to OSPF. More recently, CBT [BFC93], a routing protocol that uses shared multicast distribution trees, and PIM [DEF+94] which supports both shared and shortest path multicast distribution trees have been proposed to overcome scalability problems that are inherent in the DVMRP and PIM protocols.

In particular, DVMRP has been deployed at hundreds of sites connected by the MBONE [Eri94]. DVMRP uses the *truncated reverse-path broadcast* [DC90] routing algorithm. DVMRP differs from unicast distance vector routing protocols like RIP in an important way: while RIP thinks in terms of possible *paths* to a particular *destination*, DVMRP keeps track of the *return paths* to the *source* of multicast datagrams. Routers on the multicast network that receive multicast datagrams are said to be *downstream* from the routers that deliver the datagrams to them. Directly connected multicast routers use link-level multicast mechanisms such as Ethernet multicast to deliver multicast datagrams to downstream routers in the tree. When multicast routers are separated by arbitrary subnets that do not support network-level multicast, they communicate using tunnels as described in the previous subsection. Once a multicast datagram reaches a leaf network that has host group members, link-level multicast is used to perform delivery to the listening hosts.

The IP routing algorithm is modified so that each multicast router, R_M, informs all other multicast routers about which multicast addresses are being listened to by R_M's neighbours. Each router also calculates a spanning tree per source as shown in Figure 2.2(a). Any multicast datagram sent by a source is picked up by a designated multicast router, R_D, for the network on which the datagrams were transmitted. This router initially assumes that all multicast routers in the spanning tree that are downstream are members of the multicast group. A datagram from the source is propagated downstream to all the multicast routers including those that have joined the multicast group using IGMP, as shown in Figure 2.2(b). If an arriving datagram does not come through the interface that the router uses to send datagrams to the source of the multicast, then the arriving datagram is dropped. Multicast routers without downstream or local members send explicit *prune* messages upstream to remove themselves from the distribution tree. The net effect is a source-specific shortest path tree with the members forming the leaves of the tree as shown in Figure 2.2(c). Clearly, the cost of tree set-up is significant in terms of overall bandwidth consumed by datagrams sent by the sending host.

2.3.3 The Implications of Mobility

Throughout this thesis, the following terms are used to identify the principal network entities that are of interest in mobile networking. A *mobile node, MN*, is a node that may move through an IP internetwork. A *correspondent host, CH*, is any arbitrary host communicating with the MN. The network identified in the network component of the MN's IP address is called the *home network* of the MN. Any other network that a MN may visit is referred to as a *foreign network*.

Unicast Routing When the MN is moved from one network to another, its IP address no longer correctly encodes the network to which the MN is attached. When

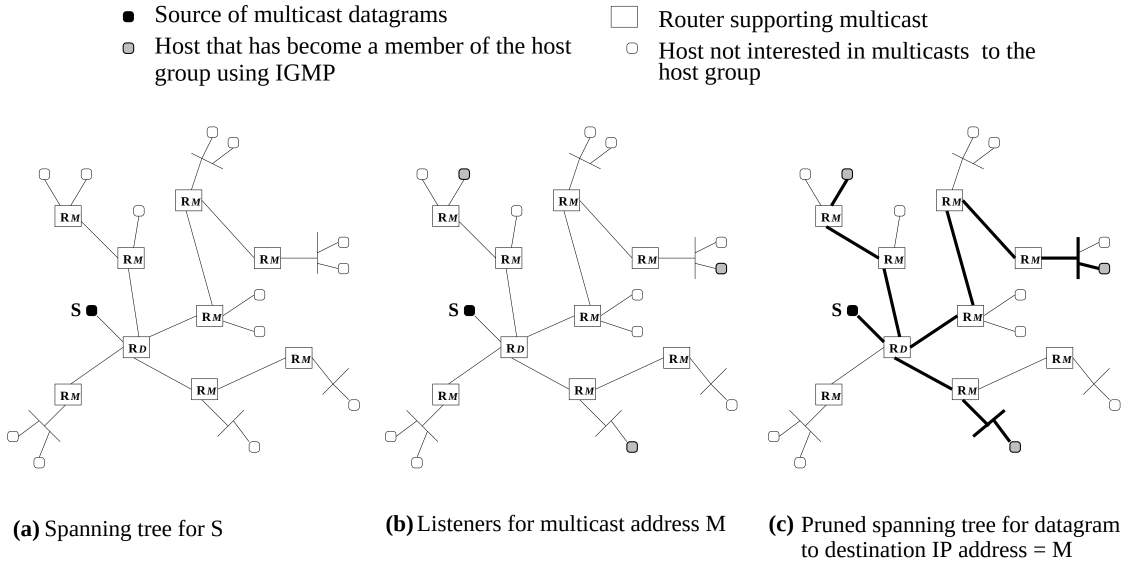

Figure 2.2: Multicast in a TCP/IP internet

a remote correspondent host, CH_R, attempts to send a datagram to the MN, the datagram is indirectly routed through the internetwork until it finally reaches a router, R_H, that attaches directly to the MN's home network. If that router still has a stale ARP cache entry for the MN, then the datagram is encapsulated in a link-level frame addressed to the MAC address of the MN and transmitted. Since the MN is no longer there to receive the frame, the datagram is effectively lost. This scenario is shown in Figure 2.3. If no ARP cache entry exists, then the router R_H sends an ARP request asking for the MN's link-level address. Since the MN is no longer present on the home network, no ARP reply is received by R_H. Thus R_H is unable to deliver the datagram from CH_R. A similar situation results if a local correspondent host, CH_L, attempts to send a datagram directly to the MN.

Multicast Routing For a MN that wishes to **receive** multicast datagrams, the routing problem is slightly different from the unicast routing problem for mobile hosts. This is because these datagrams have a Class D IP multicast address specified as their destination address. These addresses do not encode any network, and multicast routing of these datagrams can take place even if the MN is not at its home network. However, the following difficulties may be experienced if the conventional IP mul-

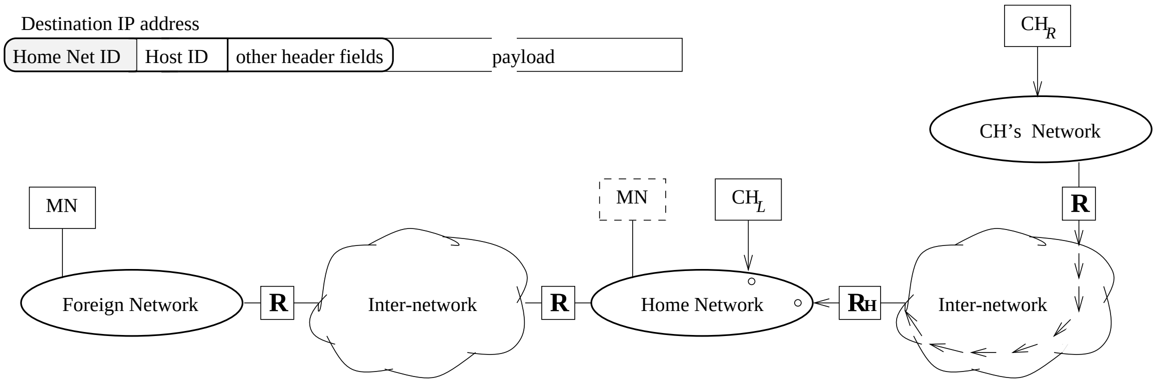

Figure 2.3: The result of mobility in an IP-based internetwork

ticast mechanism were used: the foreign network that the MN visits may not have a multicast router; and, even if the MN is able to inform a multicast router about the multicast addresses it wishes to receive, there is a setup cost associated with the establishment of a path of the multicast distribution tree to the new multicast router. Thus, when the MN moves *again*, the datagrams may not be delivered to the MN at the new network immediately since the multicast routers in the internetwork have no knowledge of this movement. Further, a fast moving MN may also leave behind a trail of multicast routers that have become members of multicast distribution trees even though there may be no members of the multicast group currently at those networks.

For a MN that wishes to **send** multicast datagrams, the situation is even more complicated since multicast datagrams are routed based on their source address. In particular, unless the MN is at its home network, multicast datagrams sent by it could possibly be dropped. This can happen because the multicast distribution tree is always established with respect to the network on which the source resides. Once the source moves, downstream multicast routers which might now receive the multicast datagrams on a different interface from that used to send datagrams to the MN (when at its home network) will drop these datagrams. Furthermore, downstream routers continue to track reverse paths to the MN assuming that the MN is on its home network. Thus the multicast routes that are established are always with reference to the MN's home network and are incorrect when the MN is at a foreign network.

2.4 Summary

This chapter has summarised issues in mobile computing relevant to this research work. To establish the context in which this research is applicable, mobile computing devices and applications were reviewed. Mobile computing devices include portable computers and embedded computers. Mobile computing applications include Management Information Systems for sales and service information networking, multi-location work, and personal services.

Unicast routing in a traditional IP network is affected by mobility when datagrams are sent to the mobile host since the mobile host's IP address has a network component that is used to perform routing across an internetwork, and the network component is no longer valid when a host moves to a new network. Multicast routing is also affected by host mobility, since multicast datagrams cannot be delivered to a mobile host or be sent by it when it moves to a new network. Additional network layer support is required to make unicast and multicast routing work for mobile hosts.

Chapter 3

Mobile Unicast Routing

Several proposals have been presented to provide unicast routing support for mobile nodes in a TCP/IP internet. This chapter presents an overview of these proposals.

3.1 Virtual Networks

A scheme known as *virtual networks* was proposed by Sunshine and Postel [SP80]. One Internet network address range is assigned to each mobile "community of interest". A MN is then assigned an IP address on the virtual network corresponding to its community of interest, and retains that address as it is relocated. Routers do not need to keep track of the location of the MNs. Instead for each physical subnetwork that supports MNs, an entity called a *forwarder* keeps track of MNs on that physical subnetwork. A correspondent host that wishes to send a datagram to a MN first needs to identify a forwarder for the subnetwork to which that MN is *currently* attached. A global dynamic database is suggested by the authors for providing MN address to forwarder mappings. Once the CH identifies an appropriate forwarder, it includes a two-part partial source route in each datagram, as an IP option. The first part of the source route specifies the address of the forwarder, and the second part specifies the address of the MN. The virtual network proposal allows a MN to retain the same network address when it relocates, but the proposal has a serious deficiency since it requires the presence of a dynamic global database to identify an appropriate forwarder. This approach would not be appropriate since the database would be queried by IP hosts that are sending packets, or alternatively IP routers that are propagating packets to the MN. These hosts or routers (potentially, all hosts and routers in the Internet) would need to be modified in order to be capable of sending packets to MNs. The solution for mobility support must localise mobility functions to only few entities. Finally, it results in different sending semantics for static and mobile hosts, and the sender has no way of knowing that the host it is sending to is mobile and hence requires a database lookup.

3.2 Columbia University Mobile IP

The Columbia University Mobile IP proposal (also known as Columbia Mobile IP and Mobile*IP, which is distinct from the IETF Mobile IP protocol referred to in the rest of this thesis) by Ioannidis *et al.* [IDM91, IM93] draws upon ideas from the work of Sunshine and Postel [SP80], and work by Deering on tunneling [DC90]. Here, a MN can retain the same IP address as it relocates, but addresses are assigned such that it is possible to identify that a given address belongs to a MN. The scheme addresses mobility primarily within a small campus internetwork. Devices called Mobile Support Stations (MSSs) are responsible for forwarding traffic to and from the MNs. There is one MSS per wireless cell and at least one per subnet. When sending a datagram to a MN, an arbitrary host must route the datagram to a MSS on the local network. That MSS either delivers the datagram itself to the MN or forwards it to the MSS responsible for the destination MN. If an MSS does not know which MSS is currently responsible for a destination, it sends a *search request* to the other MSSs. When sending a datagram, a MN routes the datagram to the MSS that serves its wireless cell. If a MN roams away from the campus internetwork, it informs one of the campus MSSs of its current location. The MSS uses tunneling to forward datagrams to and from MNs. This mobility approach is targeted primarily at mobility within a campus and roaming MNs are treated as a special case.

3.3 Virtual Internet Protocol

The Virtual Internet Protocol was proposed by Teraoka *et al.* [TCT92, TYT91, TT93]. Here, in addition to carrying the home address of a MN, a datagram header can include the current address of the MN. All datagrams sent by a MN that is away from its home network carry both source addresses. The datagram header also contains a timestamp that indicates the time at which the current address was assigned to the MN. Routers and hosts that process a datagram can examine the source address and cache the mapping between the home address and the current address. Then, when generating a datagram, if a host or router has a cache entry for the destination MN, the MN's current address can be included in the datagram header. A transit router that processes a datagram can look up the destination MN's home address in its local cache, and if necessary (based on the timestamp information) update the destination MN's current address in the datagram header. If a current address for the destination MN is not present in the datagram's header, and none of the transit routers have a cache entry for the MN, the datagram is routed to the MN's home address. If a datagram reaches the home network, a router on that network is responsible for adding the MN's current address and forwarding the datagram.

Whenever a roaming MN arrives at a new subnetwork, the MN sends a connect message to its home router, which creates or updates a cache entry for the MN. Likewise, when a MN is about to disconnect, it sends a disconnect message to the home router which then deletes the cache entry. Upon receiving a disconnect message, the home router also broadcasts the disconnect message on all directly connected networks. Any VIP-capable router that receives such a message and has a cache entry for the referenced MN deletes the cached entry and propagates the broadcast. This mechanism results in flooding of the internetwork (albeit in a controlled manner) and scales poorly. Also, if some of an internetwork's routers are not VIP-capable, it is possible that the disconnect message will not be able to reach all VIP-capable routers that have an out-of-date cache entry. Since cache entries have an associated timeout, the problem is not fatal. If an out-of-date entry causes a datagram to be routed to the wrong router, an error message is sent back to the source. Any router or host that processes the message deletes the appropriate cache entry if one is present. However, the recovery is not effective since the error message sent back to the source is not necessarily routed via the router that contains the errant cache entry.

3.4 Enhanced VIP

The VIP described in Section 3.3 was enhanced by Uehara *et al.* [UTSM93] in an attempt to circumvent some of the limitations of the original proposal. The enhanced VIP header has various control flags. In particular, it is possible to inhibit the updating of a datagram's destination at routers other than the MN's home router. Another control flag inhibits the creation of cache entries based on the source address of a datagram at routers other than the correspondent host's home router. The VIP header also includes a field to record the address of the CH or router that most recently updated the destination MN's current address in the datagram's header. A cache entry at arbitrary routers or CHs can be updated using additional control packets. This introduces a lot of security concerns. Routers will mark entries for deletion and start timers instead of immediately deleting them. If a datagram is received such that matching home and current addresses are found in the cache before the expiry of the timer, a cache invalidate message is sent to the CH or router that last updated the destination MN's current address in the datagram's header. Despite the fact that improvements were made to the ability of the original VIP with regard to the deletion of out-of-date cache entries, several problems remain unsolved.

3.5 Matsushita Proposal

Wada *et al.* [WOM92] proposed two modes of operation, one of which involved packet forwarding and another called *autonomous*. The autonomous mode, which is a new approach, requires the presence of an *autonomous supporter* on the network that the MN visits. Further, it requires software modifications to all CHs that intend to communicate with the MN. In this case, CHs can cache the care-of-address of a MN and directly tunnel datagrams to the care-of-address. Additionally, whenever the MN relocates, the *home agent* sends a location update message to the previous autonomous supporter. However, given that MNs relocate, care-of-addresses of MNs that are cached can become out of date and cause datagrams to be misrouted. The proposal requires autonomous supporters to intercept datagrams. This is complicated by the fact that care-of-addresses can be reassigned after MNs relocate. Since the autonomous supporter must selectively intercept datagrams destined for MNs that have relocated while not interfering with the MNs that currently use the same care-of-addresses, it must have a promiscuous interface on a broadcast medium. However, this is an impractical solution. When the autonomous supporter fails to intercept improperly forwarded datagrams reliably, receiving MNs must be prepared to verify the destination home address for each datagram and discard a datagram if a mismatch is detected. If the autonomous supporter is indeed able to intercept misrouted packets, it forwards the datagrams onward to newer addresses if known and to the home network if not. The autonomous supporter must then send a location update to the source of the datagram so the source can update or clear its corresponding cache entry. This proposal also lacks a timestamp mechanism for the purpose of determining which of two address mappings for a given home address is the more recent.

3.6 MIP using the LSRR Option

The use of source routing via IP's Loose Source and Record Route (LSRR) option was independently proposed by two researchers, Perkins [PR92] and Johnson [Joh94]. Here, MNs are associated with Mobile Routers that each advertise reachability to a virtual network. The Mobile Router is responsible for keeping track of the current location of each MN that has been assigned an address on that network. Each wireless cell has a transceiver, a Mobile Access Station (MAS), that defines the current location of the MN. When the MN roams into a new cell, the MN informs its Mobile Router of the IP address of the cell's MAS. The Mobile Router records this information and also informs the previous MAS that the MN is no longer present. Datagrams sent to a MN are routed to the corresponding Mobile Router. In order to forward a datagram to a MN's current location, the Mobile Router adds an LSRR option to the datagram that specifies the appropriate MAS as a transit router. When the MN replies to a CH, the MN also inserts an LSRR option in the outgoing datagram again specifying the local MAS as the transit router. In theory, when the CH receives the datagram, it must reverse the recorded route and insert it as the LSRR option in all future datagrams sent to the MN resulting in optimal routing to the MN. However, this approach overlooked the fact that LSRRs are not properly implemented in most hosts on the Internet, and significantly increases the complexity of protocol implementation on hosts. Further, where implemented, LSRR option processing is often inefficient and adds to overhead.

3.7 Network Reconstitution Protocol

The Network Reconstitution Protocol (NRP) [Int87] is a unique approach to internetwork mobility. Internet addresses that are used with this protocol are redefined to contain a unique router ID and a unique MN ID. This protocol assumes router-centric routing rather than network-centric routing. The common view of internetworking is that an internetwork is a collection of networks connected by routers, and routing is performed based on the network associated with the destination host. NRP views an internetwork as a collection of routers connected by networks, where routing is performed based on the router associated with the destination host. Datagrams are forwarded by the various routers at which a MN has previously been located, but not indefinitely. The MN is responsible for deciding which of its previous addresses are still in use by its correspondents, and to request that the routers identified in those addresses forward received datagrams directly to the MN's current location. Also, in order to communicate with MNs, a MN's network layer must be modified. The goal is to take the (source, destination) IP address pair specified by the user of the network layer at the source and provide that pair to the user of the destination. However, both the source and destination can relocate after a transport level connection is established. If it becomes necessary to rewrite the source or destination address of a datagram, the original value (as specified by the network level user at the source) is stored as an IP option in the datagram header. When a datagram arrives at its destination, if the "old source" or "old destination" options appear in the header, the network layer will restore the original values before passing the datagram to the transport layer.

3.8 IETF Mobile IP

This subsection gives a brief overview of the protocol interactions relevant to the proposed research work that take place between the functional entities defined by the IETF Mobile IP Working Group. Unless otherwise specified, all references to "Mobile IP" in this thesis refer to the IETF Mobile IP proposal.

Mobile IP [Per95] defines three functional entities: the *mobile node* (MN), the *foreign agent* (FA) and the *home agent* (HA). A MN is the node that may move through the IP internetwork. Each MN is assigned a constant IP address on a home network, which is known as the mobile node's *home address*. Correspondent Hosts (CHs) always address packets destined for a MN to its home address. When a MN connects to a *foreign* network, it identifies and registers with a FA, which it also uses as a default router. An example scenario is illustrated in Figure 3.1. The FA, during the registration process, provides the MN with a *care-of address*, which defines the MN's current location. The combination of the MN's home address and the care-of address is known as a *binding*. A binding also contains a timestamp that originates from the MN and is updated each time the MN attempts to register or re-register with a FA. The timestamp may be used to compare the age of MN bindings.

Each FA maintains a *visitor list* identifying those MNs that it currently has regis-

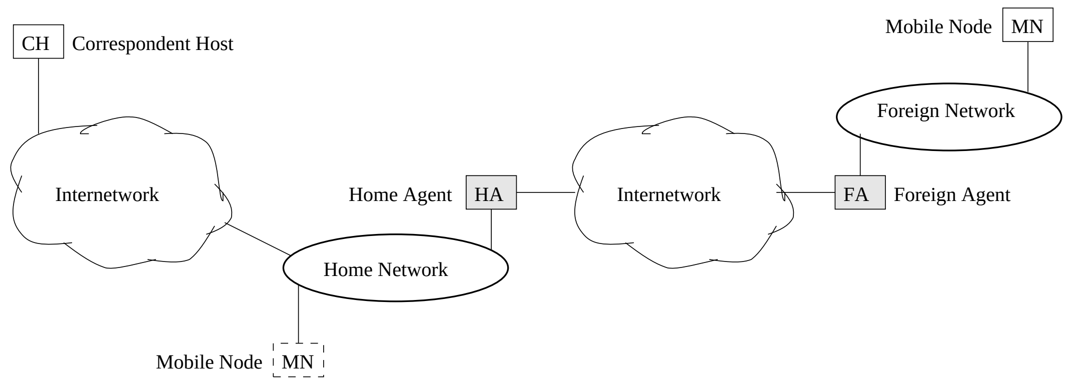

Figure 3.1: Mobile IP entities and their inter-relationships

tered. Each visitor list entry is marked to be deleted after a period negotiated during the registration process. A MN must re-register before this period has expired to ensure uninterrupted service from the FA. Each MN must have a HA, which is attached to the MN's home network. The HA maintains a *server cache* identifying all MNs it is configured to serve. A HA must always maintain a current binding for all the MNs in its server cache and thus when a MN registers with a new FA, it must also register with its HA.

A binding stored by a HA, resulting from a MN registering with the HA, is marked to be deleted after a period negotiated during the registration process. A MN must re-register before this period has expired to ensure uninterrupted service from its HA. In the case that a MN does not re-register, the home agent assumes that the MN is connected to its home network. When a MN is connected to its home network packets are delivered to it normally. If this is not the case, the HA must arrange to intercept any packets addressed to the MN. Normal IP routing mechanisms are used to route packets from CHs addressed to the MN, to the MN's home network.

When the HA intercepts a packet from a CH addressed to a MN in its server cache, it *tunnels* the packet, by *encapsulating* it and sending it to the FA that currently has the MN registered. When a FA receives a tunneled packet addressed to a MN on its visitor list, it *decapsulates* and delivers the packet locally to the MN. Otherwise, it simply drops the packet. A FA may not use a visitor list entry to deliver a packet to a MN unless the packet has been first tunneled to the FA. If the packet were not tunneled to the FA, it would have received the packet in its capacity as an ordinary IP router (assuming it also functions as one) and hence routes such a packet using the existing IP routing mechanisms.

3.9 Summary

Various proposals for performing unicast routing to mobile hosts in an internetwork have been surveyed in this chapter. The original proposal was to employ a global dynamic database for providing information on the current location of the MN. The Columbia Mobile IP proposal renewed interest in the issue of routing support for mobility. It provided an effective solution that worked well for small internetworks, but did not scale well because of a high associated search cost. Several other protocols, including the Virtual Internet Protocol, Enhanced VIP, IETF Mobile IP and others, were also reviewed.

The IETF Mobile IP protocol is in the process of being specified by an Working Group, and is being developed as an Internet standard. This protocol utilises a simple but effective approach, and is designed to scale well. It is likely to gain widespread acceptance and has hence been chosen as the foundation for the mobility support being investigated in this thesis. Throughout the remainder of this thesis, the term "Mobile IP" refers to the IETF Mobile IP proposal.

Chapter 4

Routing Subsystem Design

This chapter describes the routing subsystem designed and implemented to support mobile unicast routing in a TCP/IP internetwork. The subsystem requirements are considered first, followed by an architectural overview and an examination of the design issues involved.

4.1 Design Requirements

The support of mobile operation as specified in the Mobile IP proposal requires:

- that the HA agree to serve the MN, by inserting the MN's mobility binding in its server cache. Furthermore, the HA must intercept packets destined for the MN.
- that the FA agree to serve the MN, by inserting the MN's mobility binding in its visitor list.
- rules for the exchange and maintenance of location information.
- the actual routing of packets given that location information is already in place. This constitutes the routing algorithm. The introduction of mobility necessitates a change in the way routing is actually implemented in routers supporting Mobile IP in the Internet.

Within these four requirements, there is a logical separation between location management and the actual routing of packets. The first two requirements must be fulfilled as part of the *registration procedure* when the MN connects to a foreign network. These procedures rely on the underlying location management procedures (the third requirement). Finally, the fourth requirement actually deals with the routing of packets.

The routing system design presented in this thesis divorces the protocol used to exchange registration messages and exchange and manage location information from the algorithms that actually effect the routing of packets. This corresponds to a separation of policy and mechanism in a manner analogous to conventional routing, where protocols like RIP are responsible for the exchange and management of routing information, whereas the actual routing of packets is effected in IP itself.

The design described in this chapter deals exclusively with the actual routing of packets, while a discussion of the registration procedures and protocols for the routing subsystem presented here is deferred until Chapter 5. It is intended primarily to support Mobile IP [Per95], but it can support a forwarding mechanism for any mobile routing scheme as long as location information is made available to it as detailed in the next chapter. Further, this design is entirely in the context of implementation in the 4.3BSD UNIX operating system. The networking software in the BSD kernel is logically structured into four layers and this design confines itself to the network layer of the home and foreign agents.

4.2 System Architecture

The specification of a protocol makes no assumptions about the implementation of a network subsystem to effect it. Rather, each network entity interacts with the others, treating them as "black boxes" that conform to the same set of rules that constitute the protocol. Individual entities are free to implement the same functionality in any convenient manner. In designing the network subsystem, several issues were encountered that had to be resolved by:

- studying 4.3 BSD source code to determine the precise structuring of Mobile IP functionality so as to permit integration with the existing system. In fact, the existing 4.3 BSD source code was the single largest influence on the design presented.
- drawing from the design of other subsystem functionality in UNIX [LMKQ89], such as IP [Pos81] itself, TCP, UDP, ICMP, and conventional routing protocols [Com91] such as RIP [Mal94] and its daemon process implementation.
- studying relevant literature when available.
- resorting to intuition when no precedent existed.

4.2.1 High-level Design Issues

Three possible architectural organisations were considered:

- a completely user-level implementation: Perform datagram processing in user space, with communication using IP raw sockets [Ste90]. It is straightforward to implement the management functionality in user space.
- a completely in-kernel implementation: Perform both management functions and packet routing using kernel routines.
- a split-level implementation: Place in the kernel only those modules for which speed is critical and access to kernel routines makes kernel implementation convenient. The remaining modules would be implemented at user level, provided that access to kernel data, if and when necessary, is feasible.

The home and foreign agents are basically routing entities that must be capable of handling bursts of packets with minimal packet loss. A user-level implementation of these agents would imply a context switch overhead on every packet processed,

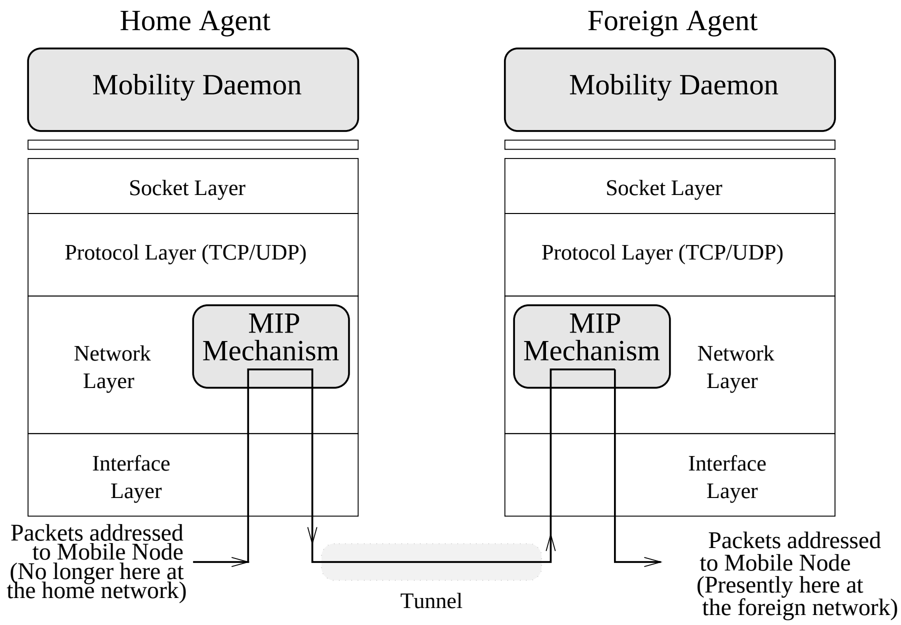

Figure 4.1: Architecture of the mobile routing software

and expensive copying to/from buffers for every packet during encapsulation and decapsulation. A kernel implementation of the routing mechanism would eliminate the context switch overhead and permit the use of UNIX *mbufs* [LMKQ89] for header manipulation. Since management packets and packets carrying location information are infrequent, it was decided to implement their processing in a user-level mobility daemon. Hence, the split-level approach was selected and the routing mechanism was implemented as shown in Figure 4.1.

4.2.2 Interface Between Management and Routing Functionality

The management and routing functionalities on the agents cooperate to provide routing support for mobile hosts. Basically, the management functionality is used to make high-level policy decisions regarding which MNs must be served and how packets must be forwarded in order to facilitate correct routing in the face of mobility.

Given that design decisions were made to implement management functionality in user-space, and routing functionality in the kernel, a means for event notification and data-exchange between the components was needed. Three mechanisms to achieve this were considered:

- Control Socket: Use a special socket to perform communication between the components. When the kernel wishes to communicate with the daemon, it forms a *control packet* and places it in the socket queue associated with the control socket. The daemon monitors this socket and performs `receive`s to read control information. Similarly, when the daemon wishes to communicate with the kernel, it forms a *control packet* and performs a `send` to write control information into the kernel. The kernel monitors the socket send queue associated with the control socket and takes appropriate actions for control information placed there by the daemon.

- Input-Output Control Operation: `ioctl` is a special system call that is used as a catch-all for performing specialised actions in the kernel. It supports a generic command interface that can be used by a process to pass information to the kernel.

- Pseudo-device: The interface between the daemon and kernel could be implemented as a pseudo-device. The interface would need to implement the standard Unix device interface calls such as `read` and `write` to exchange information.

Two considerations influenced the choice of a suitable mechanism:

- The split of functionality necessitated the passing of control information in one direction only, from the user-level daemon to the kernel.

- The desire to keep the design simple.

The `ioctl` mechanism provided enough support for inserting forwarding information into the kernel. The control socket and pseudo-device approaches, on the other hand, provided far more generality and flexibility. However, in order to avoid having to implement the somewhat more involved semantics of these mechanisms, the `ioctl` mechanism was selected. Accordingly, the system call interface of the kernel was extended by adding two new `ioctl` commands: `SIOCADDAT` to add entries into an in-kernel's Agent Table, and `SIOCDELAT` to delete entries from the Agent Table. A single table is maintained for performing both Home and Foreign Agent functions.

4.3 Design of the Routing Subsystem

Other issues encountered in the design of the Mobile IP routing subsystem are discussed in the following subsections.

4.3.1 Binding Cache Design

It is essential for the kernel to maintain enough information to make routing and tunneling decisions. The mobility bindings of MNs must be stored in

- A *server cache* for all MNs for which the agent is serving as a HA.
- A *visitor list* for all MNs for which the agent is serving as a FA.

These caches may need to be searched/modified on the arrival of management packets. Given that it was decided to make a user-level daemon responsible for the dissemination and management of location information, it was essential to store all information, including the mobility bindings and the timers associated with each, in the address space of that process. However, this information needs to be looked up for every packet that arrives in order to determine whether tunneling is necessary, making it imperative that the kernel also maintain mobility bindings in some form. This means that the daemon needs some way of updating the kernel's information whenever the location information changed, say from the expiry of a timer.

The design of the in-kernel caches as new data structures would ideally be as hash tables for efficiency, with associated routines for initialising, allocating entries, searching, entry freeing, and modifying individual fields of entries. New functionality would also need to be incorporated at the system call layer, the socket layer and the network layer itself in order to implement the operations that would be needed by the daemon to update the kernel's information. Finally, for practical purposes, the needs of testing, debugging and network management would need to be satisfied by the collection of statistics on-the-fly and software to report them in a usable manner such as by user querying.

Fortunately, the routing tables maintained in UNIX for supporting normal IP routing already exist, along with all the needed facilities mentioned previously. Figure 4.2 shows the data structures used in normal TCP/IP routing in 4.3 BSD, where the routes to destinations are stored in routing table entries. The entries store:

- a mapping from a route to a network interface structure, which corresponds to the hardware interface to be used in sending or forwarding the packet. This mapping constitutes the primary function of the routing mechanism in normal IP routing.
- information on whether the destination is directly reachable (a direct route exists) or not (it is an indirect route), indicated by a special flag. If the flag is set, at least one intermediate router exists, and this fact is used by the network-interface software to fill the link-level header using the IP address of an IP router en-route to the destination, instead of the IP address of the destination itself. The translation to a MAC address using this IP address involves a MAC address cache lookup or possible use of ARP for resolution.

For the purposes of prototyping, it was decided to reuse the existing routing tables by using an additional flag, `TUNNEL`, to indicate whether the route is to a FA (i.e., the MN is away from home) at the far end of a tunnel. Thus, server cache entries can be maintained as routing table entries with the `TUNNEL` flag set. Visitor list entries may be maintained as normal routing table entries, as direct routes to MNs.

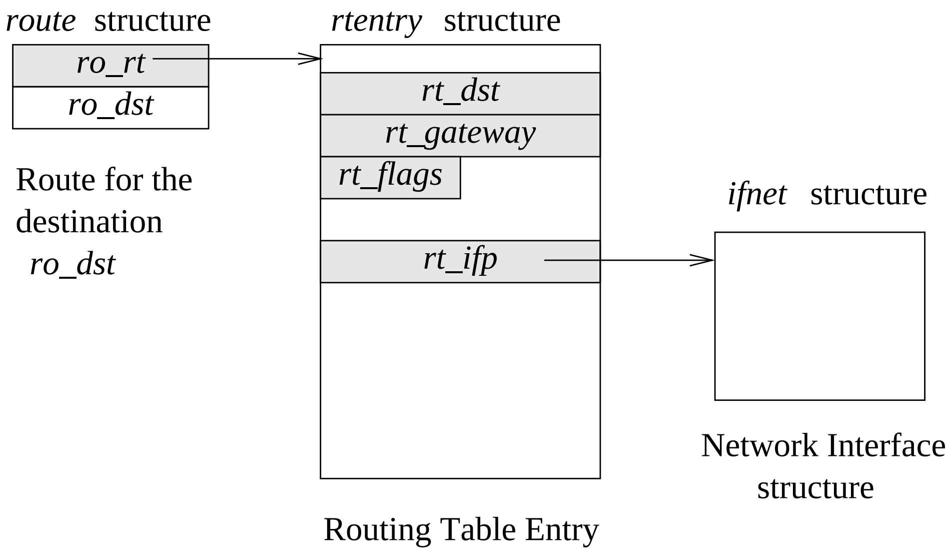

Figure 4.2: Data structures for routing

Subsequent experience revealed that unexpected interactions with normal IP routing functions were possible. To eliminate this possibility and in the interests of flexibility, forwarding information for Mobile IP has been implemented in a separate Agent Table in the kernel.

4.3.2 Demultiplexing

From among the numerous packet types that must be processed by an Agent, three types of packets are of interest: *bald* packets, *management* packets and *tunneled* packets from CHs destined for MNs.

Bald packets are addressed to the MN and are picked up by the HA's network interface. The HA uses the IP packet's destination field to differentiate between packets meant for itself, and those meant for MNs it is serving. Once identified, these packets are tunneled to the FA.

Management packets may be sent and received using UDP sockets, and require no special support since network software to hand them up to the routing daemon already exists.

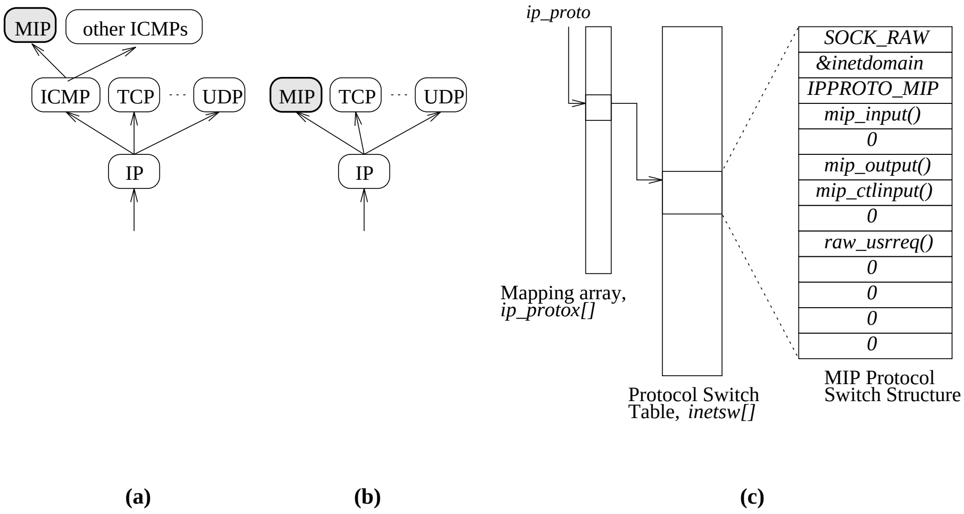

Figure 4.3: Demultiplexing and the Mobile IP protocol interface

Tunneled packets were compatible with the Internet Control Message Protocol (ICMP) in an earlier draft of the proposal, and hence, one option for demultiplexing Mobile IP packets from the rest was to define Mobile IP packets as new ICMP message types. Thus, existing software would successfully recognise them and pass them to ICMP software, inside which only trivial code would be needed to check and identify them as being Mobile IP packets as in Figure 4.3(a). This approach would allow maximum reuse of existing software, and was a viable option for prototyping. However it suffers from the serious drawback that, being ICMP messages, Mobile IP packets would never cause the generation of ICMP error messages. This is because ICMP, which is primarily an error control protocol, prohibits the sending of error messages for error messages, making demultiplexing inside ICMP impractical as a real solution. Furthermore, a new protocol type was defined for Mobile IP by the IETF Working Group late into this subsystem design. Hence, it was decided to demultiplex inside IP's input routine, in a manner similar to that used for TCP, UDP, etc. The IP protocol's mapping array was leveraged to effect demultiplexing as shown in Figure 4.3(b). As shown in Figure 4.3(c), the protocol field inside the just-arrived packet's IP header is used to gain access to Mobile IP's input routine.

4.3.3 Fragmentation

Encapsulation of a packet for tunneling increases its size. The enlarged packet may exceed the maximum transmission unit (MTU) of some link in the tunnel. This would force the router that forwards packets onto that link to fragment the enlarged packet. This was found to be a serious problem because in a TCP/IP internet, once fragmented, a packet is reassembled only at the final destination. This implied that the Mobile IP encapsulation header would only be present in the first IP fragment. All other packets would arrive without the information stored in the Mobile IP header, making proper delivery impossible.

Recent work by Claffy *et al.* [CBP93] indicates that the incidence of fragmentation depends heavily on the nature of the traffic. Encapsulating packets of interactive applications like *telnet* and name server lookups introduces no fragmentation since their packets are small. For transaction-oriented application protocols like *nntp* and *smtp*, fragmentation would still be infrequent. However, bulk transfer applications like *ftp* have large packet sizes and hence fragmentation is extremely likely. Sadly, *ftp* traffic constitutes the largest component of all traffic on the NSFNET backbone [CBP93]. Hence, fragmentation is indeed a problem and merits some thought.

Accordingly, it was decided to force re-assembly of Mobile IP packets at FAs before forwarding to the MN, to facilitate proper delivery, and at HAs to necessitate only one encapsulation header per packet from a CH to the MN. The manner of encapsulation and encapsulation header formats of the two alternative encapsulations supported by the protocol is shown in Figure 4.4.

An earlier Mobile IP draft [Sim94] also specified the use of IP-in-IP encapsulation for packets that were known to be doomed for fragmentation in the tunnel interior. This is connected with the issue of ICMP Back Propagation that is discussed in the next subsection.

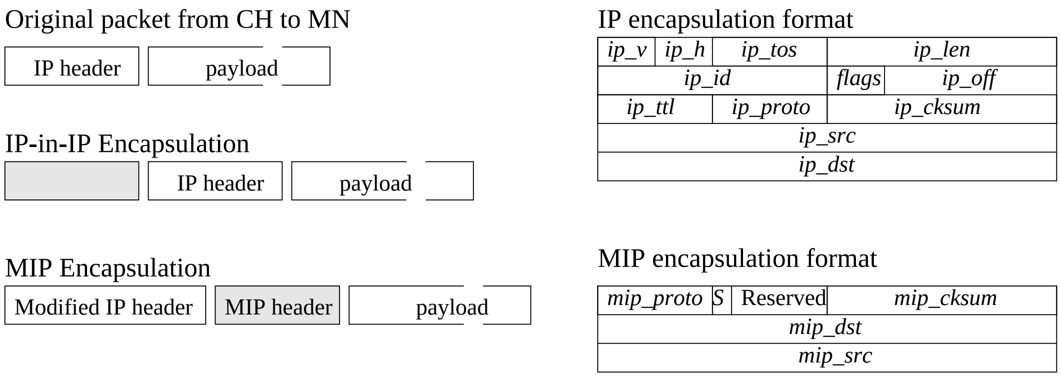

Figure 4.4: Encapsulation and the encapsulation headers used

4.3.4 ICMP Back Propagation

An important consideration in the tunneling of packets is the back-propagation of ICMP error and control messages. ICMP packets generated by routers are always addressed to the source of the IP packet that caused the error. In addition to the basic ICMP header and error-specific data, ICMP includes the first 64 bits (i.e., 8 bytes) of data from the payload of the IP packet that caused the error. Since all tunneled packets have their IP header's `ip_src` (packet source address) field set to the outgoing interface of the HA, ICMP software on all routers internal to the tunnel address their messages to the HA. Thus, all ICMP messages intended for the CH (which sent the original packet to the MN) are sent to the HA instead.

Worse still, the first 8 bytes of the IP packet causing the error contains only the first 8 bytes of the Mobile IP encapsulation header, and the `mip_src` field is not returned to the HA. Thus,

- the HA does not know to whom the ICMP message must be back-propagated.
- any back-propagated ICMP message to the CH will not contain the first 8 bytes of the IP payload of *its* packet that caused the error.

The feasibility of swapping the Mobile IP encapsulation header's `mip_src` and `mip_dst` fields was explored. This would save the source and destination address fields (i.e., `ip_src` and `ip_dst`) of the original packet. It would allow the HA to back-propagate the correct ICMP message to the CH, since the source of the original packet, (i.e., the CH's address) would be returned in the ICMP error message returned from the tunnel interior. However, this was complicated by two factors:

1. The HA would need to save the first 8 bytes of all IP payloads for a reasonable duration to facilitate faithful reconstruction of back-propagated ICMPs. This would lead to an unnecessary build up of state at the HA (in fact, the amount of such saved payload bytes would increase linearly with the amount of traffic being tunneled). Furthermore, expensive lookups would be necessary to locate a desired 8 byte sequence when an ICMP needing back-propagation did indeed arrive.

2. An earlier Mobile IP draft [Sim94] included a new IP-in-IP encapsulation, and for such packets, the 8 bytes returned by ICMP would contain the first 8 bytes of the encapsulated IP header (rather than the payload) which would not contain enough information to allow back-propagation at all.

The complications associated with ICMP messages when packets are tunneled interfere with the operation of programs like *ping* and *traceroute*. This is especially unfortunate in light of the fact that these programs are rendered useless when they are needed the most, i.e., during the deployment and testing of a new routing protocol like Mobile IP. The problem was addressed for the first time in [Sim94] where it was recommended (but not required) that the HA maintain some form of **soft state**, keeping track of ICMPs received from the interior of a tunnel. This would consist of the key attributes of the tunnel, deduced from ICMPs received, including:

- the MTU (Maximum Transmission Unit) of the tunnel
- the TTL (Time To Live), i.e., path length, of the tunnel
- the Reachability of the end of the tunnel

Using this information, the HA can return accurate ICMP messages in most cases. The HA uses the ICMP messages it receives from the interior of a tunnel to update the soft state information for that tunnel. When subsequent datagrams arrive that need to transit the tunnel, the router checks the soft state for the tunnel. If the datagram would violate the state of the tunnel (such as, the packet's TTL is less than the tunnel TTL) the HA sends an ICMP error message back to the source, but also forwards the datagram into the tunnel. Using this technique, the ICMP error messages sent by HAs will not always match up one-to-one with errors encountered within the tunnel, but they will accurately reflect the state of the network.

Support for maintaining soft state has been incorporated into the design of the subsystem. It is enlightening to note, however, that soft state does not solve the problem, but serves only to ameliorate it since the HA does not keep track of tunnel information with respect to each intermediate router inside the tunnel. Hence, *ping*s may be used to return the reachability of the MN, but will not be able to return round-trip times. Similarly, *traceroute*s to the MN would be blanked out inside the tunnel. ICMP back-propagation thus remains a limitation of Mobile IP.

4.3.5 Maintaining Tunnel Soft State

Tunnel soft state can be maintained at the user level, and handled inside the routing daemon, which can listen to an ICMP raw socket [Ste90]. However, the protocol must be capable of sending ICMP messages to the CH based on the soft state for packets in a timely fashion. Passing every packet destined for a MN to user space for verification against the state seems unreasonable, and so this approach was abandoned.

Soft state is maintained on a per-tunnel basis for all active tunnels in a list of *tunnel status blocks* (TSBs) inside the kernel as shown in Figure 4.5. The design must take into account the fact that several MNs serviced by a HA may be at the same care-of address (at the same foreign network). The soft state is updated with every ICMP message returning from the tunnel, regardless of which MN the packet causing the error was addressed to. For every packet that needs to be tunneled, the list of TSBs must be looked up. A naive search would involve chaining through the list,

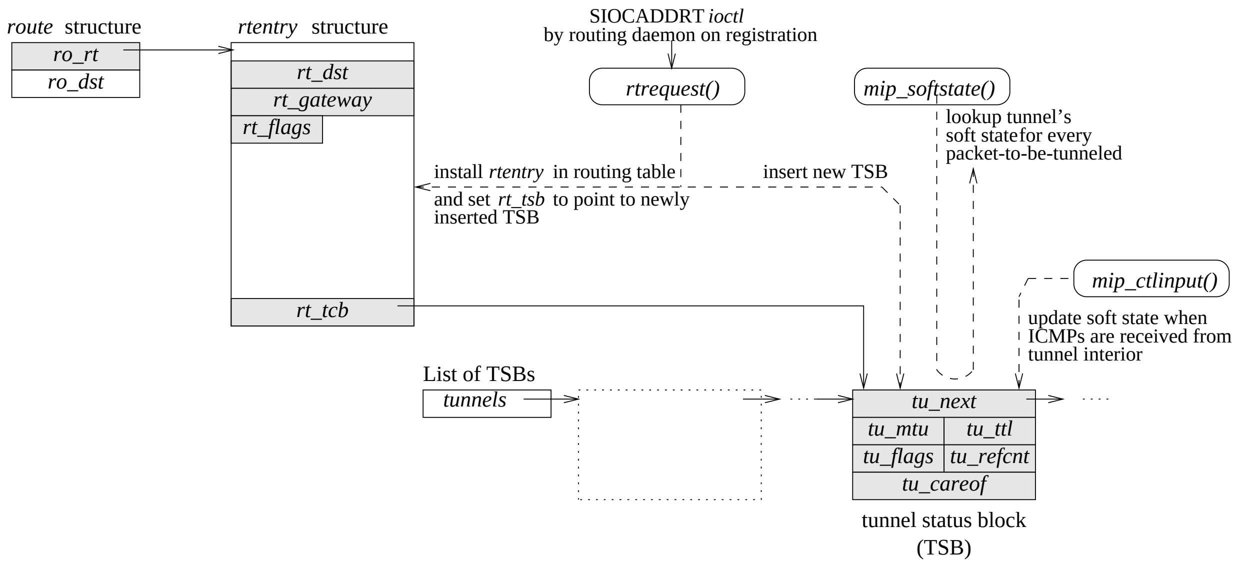

Figure 4.5: Storage and retrieval of soft state

comparing the `at_agent` field (the field storing the care-of address) of the `atentry` for the MN with the `tu_careof` (the care-of address that uniquely identifies a tunnel) field of every TSB. Because of the expense involved, it was decided to add a new field, `at_tsb`, to the Agent table entry that points directly to the TSB in question. This would allow all information pertaining to mobile routing to be obtained by a single Agent table lookup in IP's input routine.

Multiple MNs may be at the end of the same tunnel, and they are registered and de-registered independently. To permit creation and disposal of TSBs, a reference count `tu_refcnt` is maintained, which must be incremented with every new registration and decremented with every registration expiry. MN-FA bindings are inserted into the Agent table at registration time and deleted upon timer expiry by the mobility daemon through the `SIOCADDAT` (to add an Agent table entry) and `SIOCDELAT` (to delete an Agent table entry) `ioctl`s. The Agent table `ioctl()` interface implements these by calling the `atrequest` routine. A flag, `AT_HA` is used to indicate that the entry in the table corresponds to a MN for which the Agent is acting as a Home Agent. Hence, the list of TSBs is searched if the `AT_HA` flag is set in the Agent table entry, `atentry`, being added. On a `SIOCADDAT` operation, if a TSB is found, the reference count, `tu_refcnt`, must be incremented, and if no TSB is found, a new TSB must be created and inserted into the list of TSBs. On a `SIOCDELAT` operation, the reference count must be decremented and the TSB must be deleted when the count reaches zero. In the case of the `SIOCADDAT` operation, the new Agent table entry's `at_tsb` field must be set to point to the (possibly, just created) corresponding TSB.

This organisation of tunnel soft state enables:

- fast lookups when tunneling packets to the MN
- fast updates when ICMPs arrive back to the HA from the tunnel interior
- TSB list processing only during registration and registration expiry of MNs

4.3.6 Packet Routing Paths

The principal modules involved in mobile routing are shown in Figure 4.6, along with the specific paths taken by packets depending on whether the agent is tunneling the packet (i.e., functioning as a HA), or processing already tunneled packets and forwarding locally (i.e., functioning as a FA). The solid directed lines in the figure denote the path taken by packets being forwarded using conventional IP routing.

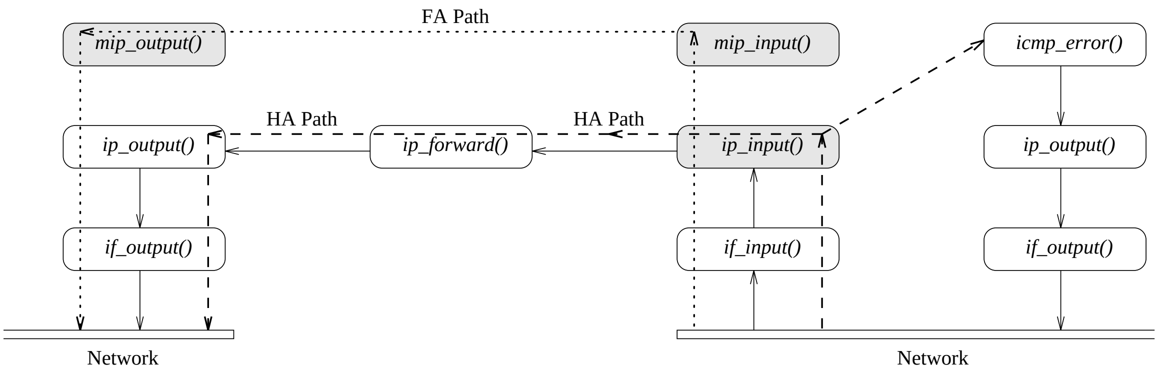

Figure 4.6: Paths taken by packets through the routing mechanism

4.4 Summary

This chapter provided a detailed look at the design of the routing functionality required to provide unicast routing support for mobile hosts. The role of the routing subsystem was first examined in the context of the Mobile IP protocol. The architecture of the subsystem was described, followed by details of the design issues that came up with regard to the binding caches, demultiplexing and fragmentation. The issue of ICMP back-propagation and the tunnel state mechanism used to deal with it were also discussed.

Chapter 5

Mobile IP Protocol Interactions

The goal of this chapter is to describe Mobile IP protocol interactions in some detail, and to specify the high level design of software modules to incorporate this protocol into 4.3 BSD UNIX. The design issues and choices that came up, and the rationale for the choices made, are discussed. The design is presented in the form of block diagrams and brief textual explanations. This design has served as a template for implementation.

5.1 Protocol Requirements

The support of mobile operation as specified in the Mobile IP proposal involves:

- *Movement Detection*: discovering that the MN has changed its point of attachment to the internetwork.
- *Agent Discovery*: the identification of a suitable FA at the foreign network that a MN may be visiting.
- *Registration Procedures*: the registration of the MN with the HA to enable routing to the current location of the MN, and deregistration when mobile routing is not needed.

- *Packet Interception*: the interception of packets addressed to the MN by the HA.
- *Packet Routing*: the routing of packets addressed to the MN to the FA.
- *Packet Delivery*: the delivery of packets to the MN by the FA.

In the following sections, each of these actions is examined at the level of detail necessary to put the architecture in perspective. The discussion of Movement Detection is deferred until after the section on Agent Discovery because of dependencies between these actions.

5.2 Agent Discovery

Whenever a MN changes its location and attaches to a new network, it must take steps to ensure that packets are correctly routed to it. Two possible scenarios may arise:

- the MN has moved to a new foreign network. In this case, the MN must initiate a Registration Procedure to ensure that it continues to receive packets addressed to it.
- the MN has returned to its home network. In this case, the MN must perform a Deregistration Procedure to ensure that packets addressed to it are received by it directly, without the aid of the agents.

In either case, the MN must first locate the agents on the network to which it has attached. In order to communicate with an agent, the MN must learn either the IP address or the link address of that agent. If the MN is on a foreign network, this address is also used as a care-of address and must be communicated to the HA.

Agent identification can be accomplished by storing agent addresses in a configuration file. On multicast and broadcast links, a MN can discover the network to which it is attached by listening to traffic on the link. The MN can then select the agent with whom the registration or deregistration procedure must be carried out by scanning its configuration file. This method has two serious drawbacks:

- configuration files must be maintained manually (a significant administrative burden in itself) and this would make it impossible to track dynamic changes in router availability
- the configuration file must be set up to include support for all the networks to which the MN might roam

Mobile IP specifies an alternative agent identification method that eliminates the need for manual configuration of agent addresses. The proposed mechanism described in [Per95] is an extension to the ICMP Router Discovery protocol [Dee91]. The Router Discovery protocol is an extension of the Internet Control Message Protocol (ICMP) to enable hosts attached to multicast or broadcast networks to discover the IP addresses of their neighbouring routers.

5.2.1 ICMP Router Discovery

The ICMP router discovery messages are called *Router Advertisements* and *Router Solicitations*. Each router periodically multicasts a Router Advertisement from each of its multicast interfaces, announcing the IP address(es) of that interface. Hosts discover the addresses of their neighbouring routers simply by listening for advertisements. When a host attached to a multicast link starts up, it may multicast a Router Solicitation to ask for immediate advertisements, rather than waiting for the next periodic ones to arrive. Any routers that subsequently start up, or that were not discovered because of packet loss or temporary link partitioning, are eventually discovered by reception of their periodic (unsolicited) advertisements.

The router discovery messages do not constitute a routing protocol: they enable hosts to discover the existence of neighbouring routers, but not which router is best to reach a particular destination. If a host chooses a poor first-hop router for a particular destination, it should receive an ICMP Redirect from that router, identifying a better one.

A Router Advertisement includes a *preference level* (computed by the router based on its resources etc.) for each advertised router address. When a host must choose a default router address (i.e., when, for a particular destination, the host has not been redirected or configured to use a specific router address), it is expected to choose from those router addresses that have the highest preference level.

A Router Advertisement also communicates a *lifetime*, specifying the maximum length of time that the advertised addresses are to be considered as valid router addresses by hosts, in the absence of further advertisements. This is used to ensure that hosts eventually forget about routers that fail, become unreachable, or stop acting as routers.

5.2.2 Mobile IP Agent Discovery

Mobile IP extends the ICMP Router Discovery protocol to perform Agent discovery. Agent discovery may be used by all agents, whether or not they also function as conventional IP routers. ICMP Router Advertisements that are used for agent discovery use packet *extensions*, including a *Mobility Extension*, to indicate that the router is a mobility agent. A MN may send a solicitation until the solicitation is answered by a mobility agent, and the MN can finally initiate the Registration Procedure. ICMP router advertisements that carry extensions can be identified by examining the number of advertised addresses. When the IP total length indicates that the ICMP message is longer than needed for the number of addresses present, the remaining data is interpreted as extensions.

Upon receipt of an agent advertisement, a MN compares the router address to that of its HA. If there is an exact match, the MN is at home. Otherwise, a care-of address is chosen from among advertising agents in the same fashion as the MN would choose a first-hop router. The highest preference router address on the network to which the MN is attached is used for the care-of address.

5.2.3 Design Considerations

Although ICMP is considered to be part of the Network Layer, ICMP packets are carried in IP datagrams. Agent Discovery could be implemented with the rest of ICMP functionality in the kernel, but a user-space approach was selected to keep the design easy to implement and modify. In Unix, an ICMP raw socket [Ste90] can be used to send and receive ICMP packets without intervening transport-layer packet encapsulation.

5.3 Movement Detection

The MN must be capable of detecting a change in its point of attachment. Mobile IP recommends that the MN use indications such as changes in interface status and changes in cell or administration to detect movement. The mechanisms are specific to the particular network access technologies, and thus are outside the scope of the Mobile IP protocol itself.

The Agent Discovery mechanism relies on periodic reception of Agent Advertisements by the MN. If an Advertisement is not heard by the MN for a pre-configured interval, even after a Solicitation has been transmitted by the MN, it can be concluded that the MN has lost contact with all agents. In this state, the MN may well conclude that it is either in the process of being moved or has moved to a network that has no agents. If, on the other hand, the MN receives an Advertisement that suggests that it has moved (based on a differing network component of the agent's IP address), then the MN can conclude that it has moved and its point of attachment has successfully changed. While a link-level mechanism to detect movement would be more efficient and accurate, this mechanism was chosen for its simplicity and the fact that the movement detection is not central to the mobility aspects of the Mobile IP protocol.

5.3.1 Design Considerations

The structure of the Agent Discovery and Movement Detection components are closely coupled. The MN is assumed to be either connected to a network or disconnected. A user-level process is used to monitor the network for Agent Advertisements. If no Advertisement is heard within a configurable timeout period, the MN assumes that it is moving, and transmits Solicitations at regular intervals. When it hears an answering Advertisement, it uses the sending Agent's IP address to determine whether it has moved. If it has not moved, it does nothing. If it has moved, and the sender of the Advertisement is the MN's HA, then it initiates Deregistration. If the sender is not the HA, the MN initiates Registration.

5.4 Registration Procedure

The Registration procedure exchanges information between a MN and its HA. All messages involved in the Registration procedure use the User Datagram Protocol (UDP) as the transport mechanism.

5.4.1 Registration when at a Foreign Network

The FA acts as a relay between the MN and the HA. The Registration function involves the exchange of four messages as shown in Figure 5.1.

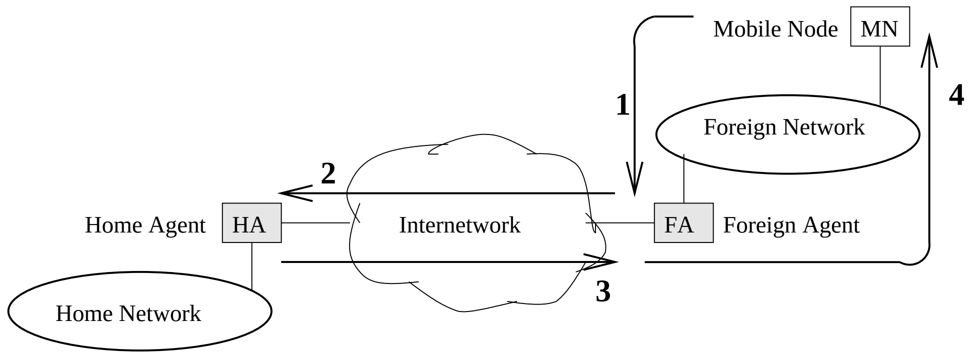

Figure 5.1: Registration function when the MN moves to a foreign network

1. The MN sends a registration request to the prospective FA to begin the registration process.

2. The FA relays the request to the HA, asking the home agent to provide the requested service.

3. The HA sends a registration reply to the FA to grant or deny service.

4. The FA relays the registration reply to the MN to inform it of the disposition of its request.

Sequence numbers are used to uniquely identify registration requests.

5.4.2 Deregistration when at the Home Network

When a MN returns to its home network, it no longer needs the forwarding service from its HA. The Deregistration function involves the exchange of two messages as shown in Figure 5.2.

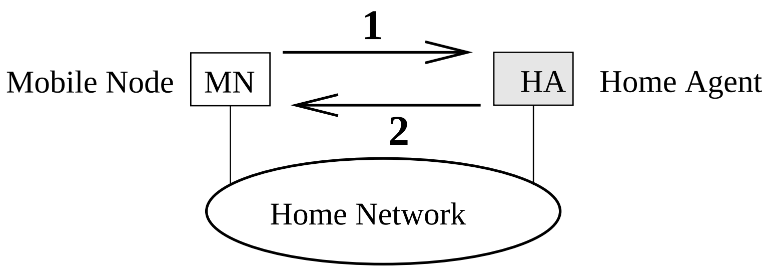

Figure 5.2: Deregistration function when the MN returns to the home network

1. The MN sends a Deregistration request directly to its HA, with a code to indicate that the HA must remove all state related to the MN.

2. The HA sends a Deregistration reply to the MN to indicate the success or failure of the MN's attempted deregistration.

5.4.3 Design Considerations

The Registration Procedure can take an arbitrarily large amount of time, depending on the latency of the internetwork that separates the FA and HA, and the availability of the MN's HA. This procedure is triggered by the Movement Detection mechanism on the MN. When the Registration Procedure has been initiated but has not been completed, it is possible that the MN may move again. In this case, the MN must abort the old Registration Procedure and initiate a new one. Since the MN would need to listen simultaneously for Advertisements and for the Registration message without blocking, the Registration Procedure was designed to run as a separate process executed upon detection of movement. If the Registration Procedure needs to be aborted, the old process is killed and a new one is executed.

5.5 Packet Interception

The mechanism used for intercepting packets addressed to the MN depends on the topological configuration of hosts and routers on the home network of the MN. Subtle issues can complicate the packet interception mechanism as will be seen in this section.

5.5.1 Home Network Configurations

There are two ways in which a home network that supports MNs can be configured:

- MNs can be placed on the same link as non-mobile hosts and the HA can itself be an ordinary host with a single interface on the MN's home network, as shown in Figure 5.3.

- MNs can be allocated a separate IP address block and placed on a network dedicated exclusively for MNs as shown in Figure 5.4. The HA can then be set up as an IP router which advertises reachability to that network using standard IP routing protocols.

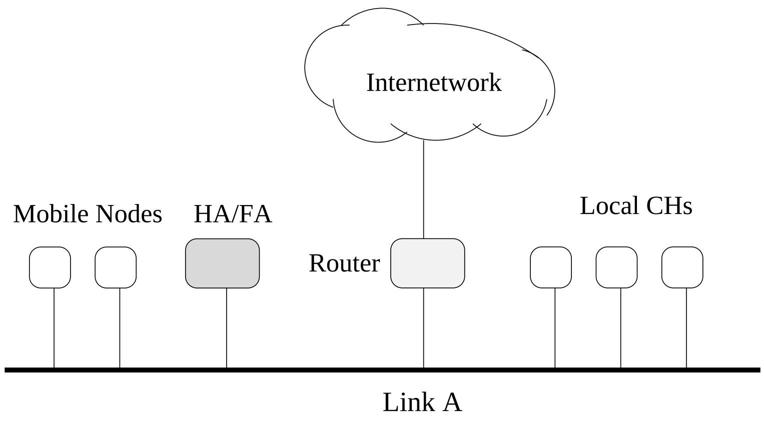

Figure 5.3: Network configuration with MNs on the same network

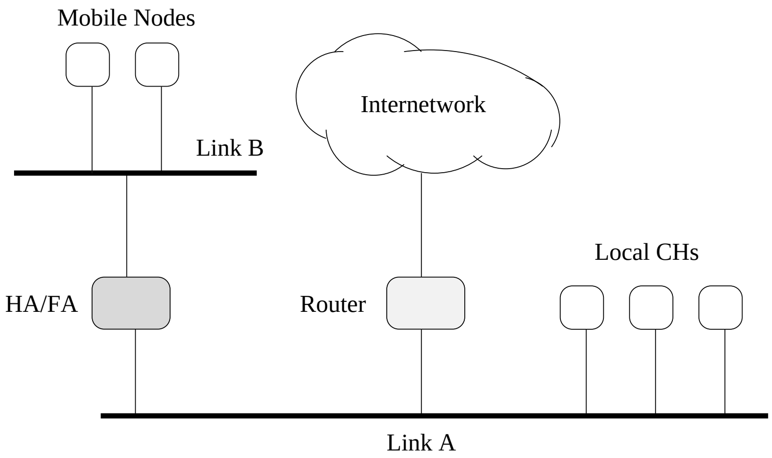

Figure 5.4: Network configuration with MNs on a separate network

With reference to Figures 5.3 and 5.4, packets addressed to the MN may be forwarded onto *link A* by a router, *Router* (if the packet's source is somewhere in the internetwork), or by a Correspondent Host, *CH*, that is directly attached to *link A*. The topology used and the physical location of the sender of a packet addressed to the MN both affect the manner in which the HA can perform packet interception.

5.5.2 Interception Mechanisms

There are two ways by which the HA can intercept packets addressed to the MN:

- Packets addressed to a MN on the dedicated network as in Figure 5.4 must pass through the HA as a consequence of normal IP routing. Thus, interception is easily achieved for such packets since they are received by the HA anyway. However, this configuration would require a small network to have an extra router between the mobile and non-mobile nodes, which can be a significant expense.

- Some packets do not pass through the HA by default IP routing. For instance, in Figure 5.4, packets that originate from other MNs on a dedicated MN network do not traverse the HA. Similarly, if the HA is not an IP router as in Figure 5.4, then no packets sent by any entity on the home network traverse the HA. To handle such cases, a technique called *Proxy ARP* [Per95] must be used for packet interception. Proxy ARP allows one host to serve as a Proxy for another, providing IP-to-Ethernet address mappings when an ARP query is seen.

In order to perform Proxy ARP, the HA responds to ARP requests for resolving the MN's IP address to an Ethernet address with its own (i.e., the HA's) Ethernet address. This results in all directly connected nodes on the MN's home network mapping the MN's IP address to the HA's MAC address.

When Proxy ARP is used, a problem can arise if a MN which has previously answered an ARP Request moves away from the link, leaving behind a stale entry in another node's ARP cache. For example, in Figure 5.3, if *Router*, which forwards datagrams onto the home network from all CHs in the internetwork, has a stale ARP cache entry for the MN, any datagrams arriving through that router for the MN will be lost. This problem is addressed in the following subsection.

5.5.3 Gratuitous Proxy ARP

From the preceding discussion, when Proxy ARP is used, it is important that ARP caches of nodes populating the home network be updated as soon as possible.

A *Gratuitous ARP* is an ARP Reply that is broadcast to all nodes on a link, but not in response to any ARP Request. When an ARP Reply is broadcast, all hosts are required to update their local ARP caches, whether or not the ARP Reply was in response to an ARP Request they had issued. A Gratuitous Proxy ARP is issued by the HA on behalf of a MN whenever the HA receives a valid registration. This causes the remaining nodes to immediately associate the home address of the mobile node with the MAC address of the HA which is now serving the MN.

While the MN is away from its home network, the HA performs Proxy ARP Replies for the MN. When a MN returns to its home network, a Gratuitous ARP should be issued on the home network to revert the address mapping from the MN's IP address to the MN's MAC address instead of the HA's MAC address.

5.5.4 Design Considerations

The ARP cache is implemented in the kernel. Changes can be made to this cache using the Input-Output Control operations (`ioctl`s) implemented in the Unix system call interface for this purpose. The SIOCSARP `ioctl` is used to set an ARP cache entry and the SIOCDARP is used to delete an ARP cache entry. Typically ARP cache entries are timed out. When Proxy ARP entries are inserted into the kernel of the Agents as part of a successful Registration, they are specially marked as being permanent.

The standard Unix ARP `ioctl` mechanism does not support Gratuitous ARP. To effect Gratuitous ARP, the kernel was modified to accept specially marked ARP cache insertion requests. At the time these entries are inserted, the kernel forms a Gratuitous ARP Reply and broadcasts it, which serves to update any stale ARP cache entries in all directly connected hosts.

When the MN returns to the home network, a Gratuitous ARP must be issued to revert to the normal mapping. The Mobile IP draft [Per95] had required that this be performed by the MN prior to initiating Deregistration. Implementation experience gained in the course of this thesis work led to the conclusion that it is impractical for the protocol to require the MN to perform this since it required access to interface-level source code or a special network device interface, both of which are often unavailable. By requiring the HA to send the Gratuitous ARP upon return of the MN, it became possible to restrict *all* kernel modifications to the Agents.

5.6 Summary

This chapter examined the protocol interactions of Mobile IP. Mobile IP requires software functionality:

- on the MNs to perform Movement Detection, Agent Discovery and the Registration Procedures,
- on the HA and FA to perform the Registration Procedures, and
- on the HA to perform Packet Interception.

The extensions to ICMP Router Discovery to perform Agent Discovery and the techniques of Gratuitous Proxy ARP were described. The procedures described in this chapter define the control and management functions of the Mobile IP Protocol and have been primarily implemented in user-level processes. In the case of the Registration Procedures and the Gratuitous Proxy ARP mechanisms, interactions with the kernel are through an input/output control operation interface.

Chapter 6

Router Evaluation

The purpose of this evaluation was to verify the correct operation of the routing functionality, and to measure its performance in order to establish the practicality of the design approach in the context of existing networking technologies. The routing subsystem that was implemented was subjected to simple tests in the Distributed Systems Performance Laboratory in the Department of Computer Science to verify successful operation of the routing functionality. Measurement experiments were conducted to gain insight into the performance of the routing subsystem. This chapter describes the tests, the issues involved in conducting the measurements, and the results of the measurements.

The Mobile IP software that was implemented consists of user-space components and kernel-space components. The user-space components deal principally with management and control information, and their performance depends on factors such as internetwork propagation delay, congestion, characteristics of wireless links, and so on. Fortunately, none of these components have stringent performance requirements, and consequently their performance is not critical. However, the kernel-space components are critical to the performance of the network support. Poor performance can make an agent a bottleneck for packets sent to a mobile host. Furthermore, higher level protocols may need to be adapted to special conditions that may occur in the network layer such as packet loss. Hence, an idea of the performance of the mobile routing functionality is essential.

6.1 Testing the Routing Subsystem

Simple tests were conducted using a passive network monitor to demonstrate the kernel routing subsystem functionality on the agents. These tests were designed to demonstrate:

- the ability of the HA routing subsystem to
  - intercept IP packets addressed to the MN (which is presumably connected to a foreign network).
  - encapsulate the packets and tunnel them to the FA.
- the ability of the FA routing subsystem to
  - demultiplex encapsulated packets received via a Mobile IP tunnel from a HA.
  - decapsulate the tunneled packets and forward them locally (to the MN which is presumably located on the FA's network).

The testing assumes that tunneling across a wide-area network is equivalent to tunneling across a LAN. In both the wide- and local-area cases, the tunneled packets use IP-in-IP encapsulation and are routed using ordinary IP. IP uses standard mechanisms to route and deliver encapsulated packets across an arbitrary internetwork. The tests are sufficient to demonstrate the routing subsystem functionality since IP can indeed be relied upon to perform packet routing and delivery across an internetwork.

Four hosts (Skippy, Darlene, Cubby and Tommy) were set up as shown in Figure 6.1. Skippy and Cubby are Sun workstations, Darlene was a M68040 processor-based HP workstation, and Tommy was a M68030 processor-based HP workstation. The setup can be described as follows:

- *128.233.128.200, MN*: as the Mobile Node, a fictitious host with a non-existent IP address whose network component is the same as the campus network address.

- *Skippy, CH*: as the Correspondent Host, which transmits ICMP echo-request datagrams (*ping*s) addressed to the MN. Note that in all the experiments, the *ping* itself fails because the host being *ping*ed is non-existent.

- *Darlene, HA*: as the Home Agent, which captures the ICMP *ping*s to the absent MN, encapsulates them and tunnels them to the FA.

- *Tommy, FA*: as the Foreign Agent, which receives tunneled datagrams from the HA, decapsulates them and retransmits the original *ping*s on the local network.

- *Cubby, as the network monitor*: running *tcpdump* (which is a network monitoring tool available on Unix systems) to capture the traffic between the communicating Mobile IP entities as seen on the local network.

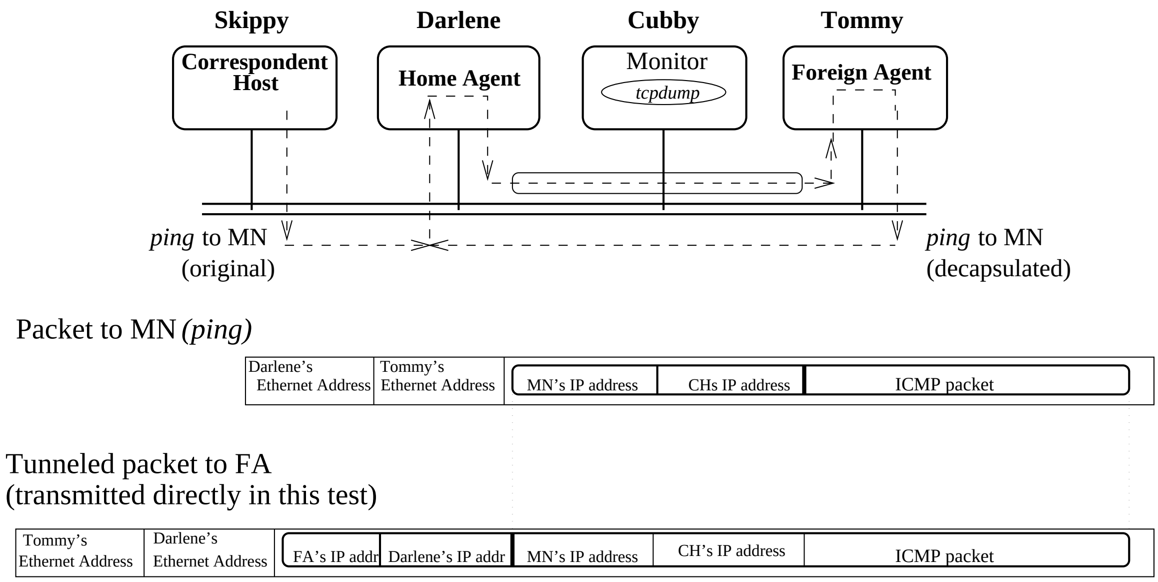

Figure 6.1: Experimental setup for network tests

The original packet (the ICMP *ping*) sent by the CH to the MN, and the packet after IP-in-IP encapsulation tunneled by the HA to the FA are shown along with the corresponding source and destination addresses in Figure 6.1. Each source/destination pair is shown with the destination address preceding the source address.

Three separate experiments were conducted and these are described in the following subsections. In all cases, the main registration functions were effected using user-level programs to perform the relevant insertions into the ARP cache and routing table of the HA, tasks that normally would be implemented by the mobility daemon.

6.1.1 Without Mobile IP

The experimental scenario may be described as:

1. The MN has moved, but not registered with the HA, hence,
   - Proxy ARP has not yet been performed
   - The MN's IP address has not yet been put in the HA cache
2. The CH tries to send an ICMP ping packet to the MN, which is supposed to be located on this network.

The datagrams seen on the network were as follows:

```
skippy> ping 128.233.128.200 3
no answer from 128.233.128.200

cubby> tcpdump -v (host skippy and host 128.233.128.200)
tcpdump: listening on le0

16:19:48.849118 arp who-has 128.233.128.200 tell skippy
16:19:49.856167 arp who-has 128.233.128.200 tell skippy
16:19:50.865144 arp who-has 128.233.128.200 tell skippy
16:19:51.875697 arp who-has 128.233.128.200 tell skippy
```

The CH repeatedly attempts to ascertain the MN's Ethernet address by broadcasting ARP requests that are unanswered. Hence, the transmission ends in failure.

6.1.2 After Registration, with Non-existent FA

The experimental scenario:

1. MN registers with HA, causing HA to
   - Perform Proxy ARP for the MN, i.e., respond when the CH tries to find out MN's Ethernet address. This allows the HA to grab the packet meant for the MN.
   - Insert a Routing Table entry for the MN, causing any packets meant for the MN that are grabbed, to be correctly tunneled to the FA (in this case, the fictitious agent 11.22.33.44)
2. The CH tries to send an ICMP ping packet to the MN, which is supposed to be located on this network.

The datagrams seen on the network were as follows:

```
skippy> ping 128.233.128.200 1
no answer from 128.233.128.200

cubby> tcpdump -v (host skippy and host 128.233.128.200) or
                    (host darlene and host 11.22.33.44)
tcpdump: listening on le0

17:38:13.783412 arp who-has 128.233.128.200 tell skippy
17:38:13.784419 arp reply 128.233.128.200 is-at 8:0:9:9:15:8

17:38:13.784769 skippy > 128.233.128.200: icmp: echo request
17:38:13.803375 darlene > 11.22.33.44: ip-proto-15 84
```

The CH is able to resolve the MN's IP address to an Ethernet address correctly, since the HA performs proxy ARP for the MN. Thus, the CH successfully transmits its *ping* datagram, reported by *tcpdump* as `icmp: echo request` datagram from skippy to the machine with IP address 128.233.128.200. The HA successfully captures the datagram, and tunnels it as an IP-in-IP encapsulated datagram (of protocol type 15, denoted in the *tcpdump* trace as `ip-proto-15`) to the FA, at address 11.22.33.44. It may be noted that the ICMP request which is 64 bytes long by default has been enlarged by 20 bytes (the size of a minimal IP header) to 84 bytes by the encapsulation. The encapsulated datagrams are sent by Darlene to its default gateway that attempts to route it to the host 11.22.33.44 (which may not exist since it is fictitious) through the Internet.

6.1.3 After Registration, with FA on Local Network

The experimental scenario:

1. The MN registers with the HA, causing the HA to
   - Perform Proxy ARP for the MN, i.e., respond when the CH tries to find out the MN's Ethernet address. This allows the HA to grab the packet meant for the MN.
   - Insert a Routing Table entry for the MN, causing any packets meant for the MN that are grabbed, to be correctly tunneled to the FA (in this case, Tommy)
2. The CH tries to send an ICMP ping packet to the MN, which is supposed to be located on this network.

The datagrams seen on the network were as follows:

```
skippy> ping 128.233.128.200 1
no answer from 128.233.128.200


cubby> tcpdump -v (host skippy and host 128.233.128.200) or
                 (ether host 8:0:9:9:15:8 and ether host 8:0:9:10:16:3d)
```

```
tcpdump: listening on le0

18:05:06.583114 arp who-has 128.233.128.200 tell skippy
18:05:06.584217 arp reply 128.233.128.200 is-at 8:0:9:9:15:8

18:05:06.597923 skippy > 128.233.128.200: icmp: echo request
18:05:06.599113 darlene > tommy: ip-proto-15 84

18:05:06.601455 arp reply 128.233.128.200 is-at 8:0:9:9:15:8
18:05:06.608236 darlene > tommy: ip-proto-15 84

18:05:06.609776 skippy > 128.233.128.200: icmp: echo request
18:05:06.610689 darlene > tommy: ip-proto-15 84

18:05:06.611829 skippy > 128.233.128.200: icmp: echo request
18:05:06.613063 darlene > tommy: ip-proto-15 84

. . . and so on.
```

As seen in the previous test, the ARP resolution and encapsulation take place correctly. In this case, however, the FA is a host (Tommy) that is present on the local network. The command to *tcpdump* captures all datagrams passing between the network interfaces of the HA (Darlene) and the FA (Tommy), in addition to the *ping* to the MN and associated ARP request. When the encapsulated datagram is received by the FA (Tommy), the FA attempts to transmit it on the local network. In doing so, it broadcasts an ARP request (excluded from the *tcpdump* trace) which is resolved by darlene. In a real situation this would not be so, and the MN itself would have replied with its correct Ethernet address, ensuring correct delivery to the MN. Under the setup of this experiment, conducted on a LAN, the FA decapsulates the tunneled datagram, and delivers it to Darlene's Ethernet address. Darlene, which has been set up as a HA, (again) tunnels it to the FA (Tommy), which decapsulates it and forwards it locally (again), and so on. Under the set up, the datagram circulates forever between the two hosts until either the ARP cache entry or the routing table entry in Darlene is deleted, or until either one of the agents is shutdown.

6.2 Kernel Instrumentation

The performance of the Mobile IP router implementation was evaluated by measurement. This included measurement of the delays in the IP and interface queues, the latency of the routing subsystem, and the total latency of the system. The throughput capacity of the system and lengths of the IP and interface queues were also measured. Measurements were performed using *kernel probes* as was done in [PP93]. Kernel probes provide a mechanism to instrument the operating system for the purpose of taking measurements and are described in the next subsection.

6.2.1 Kernel Probes

A kernel probe is a small segment of code placed strategically in the operating system kernel to record information during the execution of the instrumented code.

Probe Location: As shown in Figure 6.2, four queues labelled A, B, C and D lie along the path taken by packets that are routed to MNs: the Home Agent's Network queue (A), the Home Agent's Interface queue (B), the Foreign Agent's Network queue (C) and the Foreign Agent's Interface queue (D). These queues define reference points where measurements may be taken. Two probes are associated with each queue: one that records information when data is appended to the queue, and the other that records information when data is removed from the queue. When activated, each probe records a timestamp, the amount of data passing through the reference point, and a type field to identify the location of the probe. The information recorded is minimal, but can nevertheless provide enough information about the behaviour of the

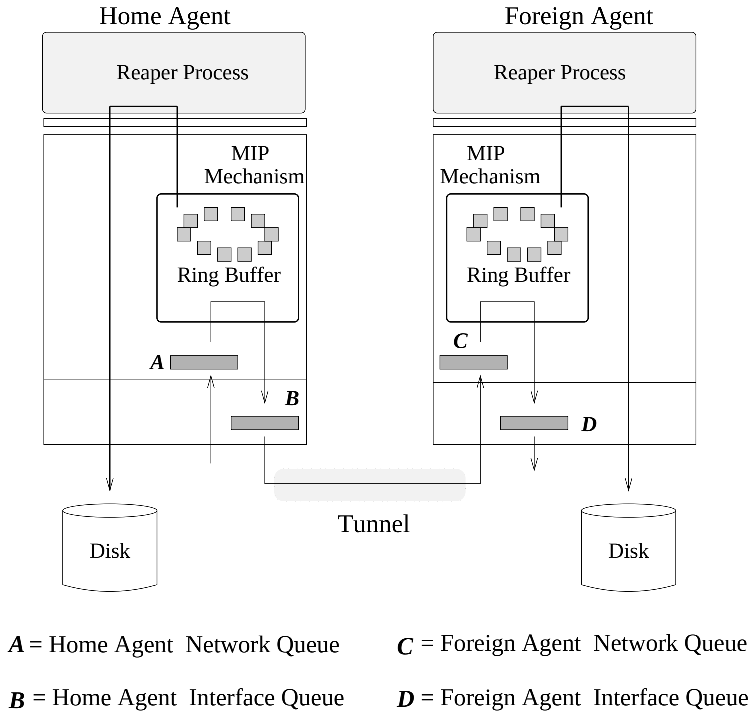

Figure 6.2: Routing subsystem measurement setup

system. For example, the difference in the timestamps recorded for a packet on either side of a queue provides the queueing delay for that packet in that queue.

Probe Overheads and Operation: The introduction of measurement functionality in the kernel can introduce overheads that might have a significant impact on the measurements. These overheads must be kept to a minimum. Three overheads are associated with the probes:

- overhead due to probe execution time: To minimise this, any kind of processing other than logging a few important parameters was avoided.
- overhead due to storing of measurements: To address this, the probes stored records in the kernel rather than in files. A ring buffer mechanism was implemented in the kernel to achieve this.
- overhead due to probe activation: The probes were activated by testing the protocol type field of the packets for the protocol type of Mobile IP, and the

encapsulation protocol, IP-in-IP.

6.2.2 Timer Considerations

The recording of accurate timestamps by the probes is key to getting credible performance figures. Hence, the choice of an appropriate routine to obtain timestamps in the kernel was important.

Unix kernels maintain the system time by using a hardware counter to interrupt the kernel at a fixed rate, defined by the kernel: for example, 50 Hz in the HP-UX kernel, 100 Hz in the SunOS kernel, 256 Hz in the Ultrix kernel and 1024 Hz in the OSF/1 kernel. The interrupt frequency on the HP 9000 workstations used in these experiments (Darlene and Tommy) thus translates to a clock resolution of 20 milliseconds. Upon each interrupt, the globally maintained *system time* is advanced by 20 milliseconds. In the HP-UX kernel, the `kernel_gettimeofday()` function provides time accurate to microsecond resolution by calling the assembly language routine `_get_precise_time`. This routine reads the hardware counter on the timer chip, and obtains the exact current time by adjusting the system time (the time of the last clock interrupt) with the hardware counter value. This mechanism was found to provide the required accuracy for the timestamps that were recorded.

6.3 Experimental Design

The design of the performance measurement experiments involved the choice of suitable factors to be varied, the implementation of the components that perform a role in the experiments, and also the design of a suitable framework to glue together the components for conducting the experiments. These issues are discussed in this section.

6.3.1 Experimental Issues

Network hardware technologies implemented in the data-link layer and physical layer of the ISO OSI model place an upper bound on the amount of data that can be placed on a physical link in one transmission. This upper bound is called the *maximum transfer unit* (MTU) of the physical network. TCP/IP specifies an IP datagram size of 64 kilobytes, and a mechanism to divide a datagram into smaller pieces when the datagram needs to traverse a network that has a MTU smaller than the size of the datagram. The small pieces are called *fragments* and the process of dividing a datagram is known as *fragmentation*.

IP fragmentation represents an overhead on routing in an internetwork. Mobile IP uses encapsulation which may force fragmentation in a significant number of packets since they may already be close to the size of average network MTUs [CBP93]. Since the use of IP-in-IP encapsulation is unavoidable, the extent to which fragmentation affects network performance is of interest.

Factors and Levels: The measurements were conducted on both the Home and Foreign Agents. In measuring the performance of the mobile routing subsystem on each Agent, the size of the packets to be routed was varied to observe performance for different packet sizes. In order to maximise the utility of the data collected during the measurement process, the following key packet sizes were considered:

- minimal and maximal (as defined by the MTU of the link) IP packet sizes.
- the largest size that did not cause the outgoing packet to be fragmented.
- the smallest size that caused the outgoing packet to be fragmented.

The minimal packet size on the Home Agent is the size of an IP header without any attached data (20 bytes). In order to account for the encapsulation, the minimal packet size on the Foreign Agent is the sum of two IP headers without any associated data (40 bytes). The maximal packet size chosen was 1497 bytes which is the Ethernet

MTU. Other packet sizes were chosen to ensure an even distribution of packet sizes over the range from 20 to 1497. The specific values chosen for the packet sizes are presented in Table 6.1.

Table 6.1: Packet sizes used in the experiments

Agent	Input Packet Sizes
Home Agent	20, 320, 620, 920, 1220, 1477, 1478, 1497
Foreign Agent	40, 320, 620, 920, 1220, 1477, 1478, 1497

Minimising Interference: Network measurements are easily affected by network traffic not related to the experiments. Since it was not feasible to use dedicated network segments for the experiments, it was not possible to completely eliminate the effects of unrelated traffic. Three steps were taken to mitigate the impact of unrelated traffic on the measurement results. First, the experiments were conducted at a time that was anticipated to have the least possible network load due to user activity (such as NFS traffic). Second, in order to minimise the effect of unrelated traffic, it was desirable to use a significant amount of raw data for computing the performance metrics. However, in order to restrict the volume of trace data collected, and to ensure a reasonable length of time for a single measurement run, only a modest volume of data could be collected. As a compromise, the data generated by 1000 packets were used for computing the performance metrics. Each *measurement run* consisted of measurements collected for a single packet size for a given Agent. Third, in order to dilute the effect of unexpected variations in the raw measurement data due to unrelated traffic, 10 measurement runs were performed for each combination, but the combination was kept fixed within each run. The final figures reported in this thesis are those obtained by averaging the results obtained across the measurement runs conducted for a given (Agent, packet size) pair.

6.3.2 Components for Trace Collection

The requirements of the experiments implied several discrete components that would need to execute in a co-operative fashion in order for the raw measurement data to be collected. These components included:

- *Probes* to be activated when the relevant packets are being processed in the router, to effect logging of necessary information.
- *packet drivers* to pump packets to be processed in the router.
- A *reaper* process to flush the logged information into trace files.

The operation of the probes has been described earlier in Section 6.2.

Packet Drivers: It was desirable to use a custom source of packets for generating the packets that trigger tracing in the Agent kernels rather than an existing source of packets, such as the *ftp* program which uses the FTP protocol to send messages in TCP packets. This was desirable in order to:

- make it easy to trigger logging by the probes. The probes in the kernel could be easily activated for only the desired packets by using distinct protocols that are not normally seen on the network.
- avoid unexpected interactions that would affect the measurements (such as acknowledgements generated for each packet).
- provide maximum flexibility in varying parameters such as the size of the packets to be transmitted.
- bypass transport layer mechanisms that would affect the flow of packets, such as TCP's window-based flow control mechanism.

Packet drivers were implemented in user space, and used raw IP sockets to enable them to construct their own packet headers. Two packet drivers (*pumpers*) were implemented:

- *pumpHA*, which transmitted packets of a new, unused protocol type to activate logging on the Home Agent.

- *pumpFA*, which transmitted IP-in-IP packets to activate logging on the Foreign Agent.

Trace Collection: Efficiency considerations in the kernel required the use of a pre-allocated ring-buffer in the kernel for maintaining the event log. However, resource limitations in the kernel dictated that only a small number (1800) of log records could be pre-allocated. Each packet generates four log records in an Agent's kernel (six, if fragmentation occurs). The amount of data recorded in the kernel is constrained by the number of log records, and data would be lost if the number of packets that trigger logging exceeds 450 (300, if fragmentation occurs). To permit larger amounts of data to be collected, a privileged user process called *reaper* is used to directly read filled log records from the kernel's virtual address space and flush them to disk, thereby making the log records available for use again. When no more log records remain to be flushed to disk, the *reaper* sleeps for a small amount of time. To ensure that recorded information was reaped before the filled log records are overwritten, only a small delay was used between successive attempts of the *reaper* process to read filled log records from the kernel. The flushed data stored in trace files on disk was then used for off-line processing and analysis.

A reliable framework was needed to synchronise the component of the network that caused measurement data to be produced (the *pumper*), the component that recorded the measurement data (the *probes*), and the component that collected the data for off-line analysis (the *reaper*). This framework, described in the next subsection, prevents measurement data from being produced faster than it can be collected and provides a flexible platform for conducting the measurement experiments in a controlled and repeatable manner.

6.3.3 A Framework for Trace Collection

The components that had to be synchronised resided on physically distinct machines. As shown in Figure 6.3, two processes communicating via a reliable TCP connection (shown as the shaded region in the figure) were used to synchronise the components and schedule the measurement runs. The trace collector process, *runreap*, was executed on the Agent on which the measurements were to be carried out. A single measurement run is described in the following paragraph and is illustrated in Figure 6.3.

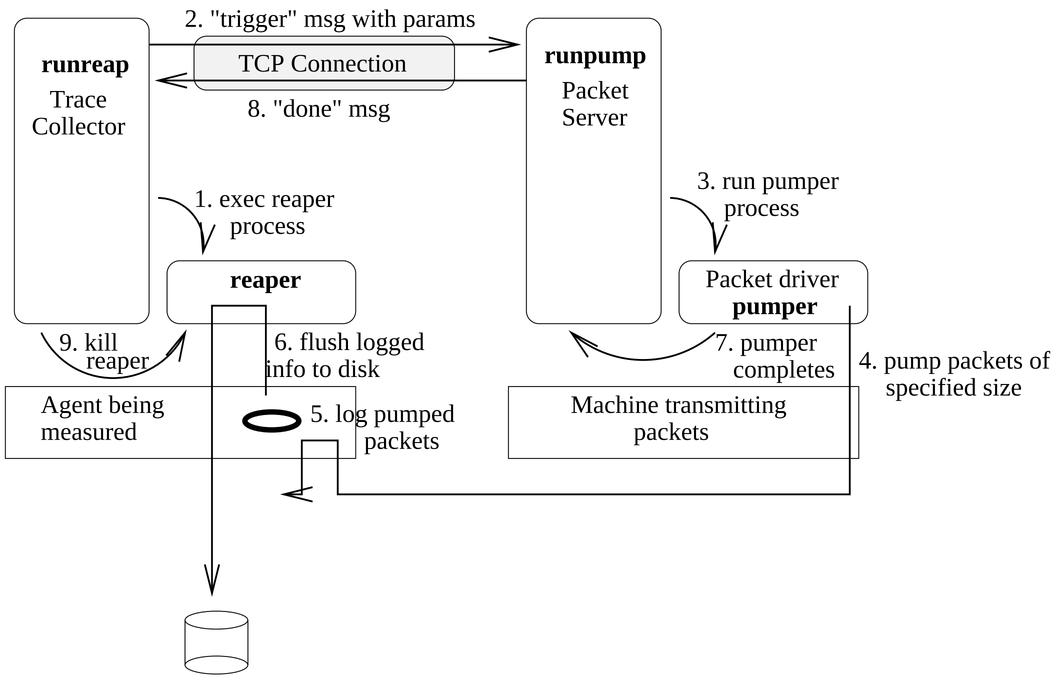

Figure 6.3: Measurement procedure and component interactions

The trace collector starts a run (1) by starting a reaper process that executes concurrently with it. Since no log records would have been filled, the reaper process blocks. It wakes up periodically to check if any records have been filled, and if so, flushes them to disk. The trace collector then sends a *trigger* message (2) to the packet server program, *runpump*, specifying the type of packets to be transmitted onto the network, the size and the number of packets, and the destination address to which the packets must be addressed. The packet server immediately runs the appropriate packet driver (3) by passing it the relevant parameters.

The packets pumped by the packet driver (4) activate the probes in the kernel of the Agent being measured, and log records are filled (5). Whenever the reaper process wakes up, it discovers filled records and flushes them to disk (6). It is assumed that interference caused by the disk activity would not cause significant interference since the disk I/Os are performed on behalf of a user process, whose execution is pre-empted by the kernel whenever network I/O must be performed.

After the packet driver has run to completion, it returns control to the packet server program (7) which sends a *done* message to the trace collector (8) to indicate that all the packets that were requested were transmitted. After waiting briefly to ensure that the reaper would have flushed all records that were filled in, the trace collector kills the reaper process (9).

The trace collector specifies the packet sizes, the number of packets to be used per run, and also the time between successive transmissions by the packet driver. These parameters are communicated to the packet server when an individual measurement run is performed. Since the Home and Foreign Agent functionalities are co-resident on the same machine, measurements pertaining to both Agents can be collected by a single invocation of the trace collector. The measurement framework designed for the purpose of these experiments is highly flexible and needs no intervention to perform the runs necessary to obtain the required measurements. The results of the measurement experiments are discussed in the next section.

6.4 Routing Subsystem Measurement Results

The log records flushed into trace files on disk as described in the previous section were processed and analysed off-line to yield the measurements for the router implementation. These measurements are described in this section.

6.4.1 Delay Measures

Delay measures were obtained by using the timestamps recorded in each log record. The *IP queue delay* corresponds to the difference of the timestamps taken when a packet is enqueued and dequeued in the Agent's IP queue. The *interface queue* delay corresponds to the difference of the timestamps taken when a packet is enqueued and dequeued in the Agent's interface queue. The *processing delays* include: the *routing delay* which is the time taken to determine the route for the input packet, *i.e.,* corresponding to the difference in the timestamps taken when a packet is dequeued from the Agent's IP queue and enqueued in the Agent's interface queue; the *total delay* which is the time taken to process a packet from the time the packet is read from the interface card, to the time the routed packet is written onto the interface card. This corresponds to the difference in the timestamps taken when a packet is enqueued in the Agent's IP queue and dequeued from the Agent's interface queue.

Home Agent Queueing Delays The queueing delays at the Home Agent's IP and interface queues are shown in Figure 6.4.

The queueing delays are almost constant with increasing packet size, increasing just a little bit when fragmentation occurs. Each packet takes about 130 microseconds to go through the IP queue and about 50 microseconds in the outgoing interface queue. The constant delay is because data copies are not made, and consequently the processing cost is largely insensitive to packet size. The small increase at a packet size of 1478 bytes is due to the increased processing time involved in fragmentation, which increases the delay in service for packets queued in the IP queue.

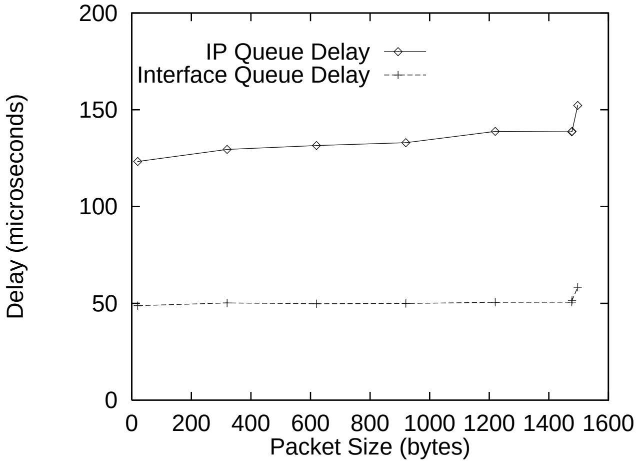

Figure 6.4: Queueing delays on the Home Agent

Home Agent Processing Delays The processing delays through the Home Agent are shown in Figure 6.5. Clearly, the routing delay (about 500 microseconds) accounts for most of the total delay (about 700 microseconds) with the queueing delays in Figure 6.4 accounting for the remainder (around 200 microseconds).

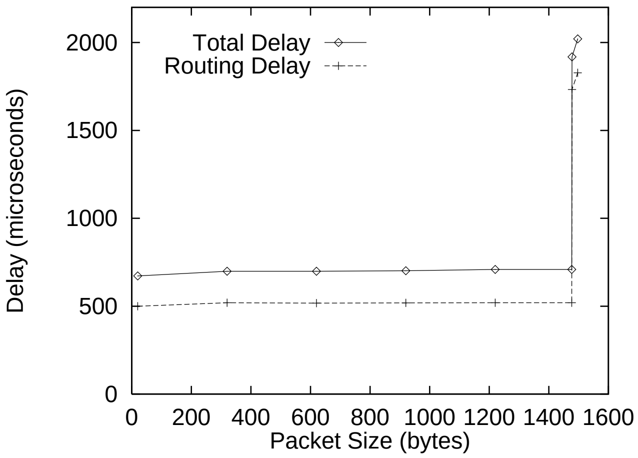

Figure 6.5: Processing delays on the Home Agent

The processing delays on the Home Agent are almost constant until the packet is large enough to require fragmentation. Fragmentation imposes additional overheads for performing header allocation and processing, making a physical data copy, and performing two network transmissions instead of one. As shown, this adds considerable delay (from 700 microseconds to 1900 microseconds).

Foreign Agent Queueing Delays The queueing delays at the Foreign Agent's IP and interface queues are shown in Figure 6.6.

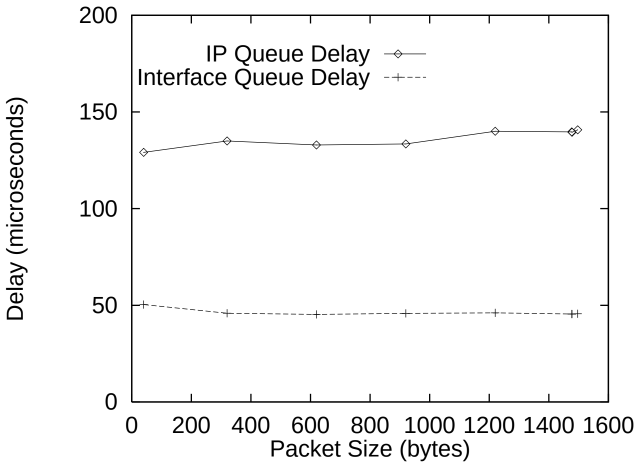

Figure 6.6: Queueing delays on the Foreign Agent

The queueing delays are found to be almost constant with increasing packet size. The queueing delays on the FA match very closely with those for the HA (130 microseconds in input IP queue, 50 microseconds for outgoing interface queue).

Foreign Agent Processing Delays The processing delays through the Foreign Agent are shown in Figure 6.7.

The processing delays on the Foreign Agent are almost constant for all packet sizes, since no fragmentation occurs (fragmentation can only happen when a packet is being encapsulated, not decapsulated). The Foreign Agent processing delays are slightly lower than those at the Home Agent (450 versus 500 microseconds).

6.4.2 Queue Lengths

Queue lengths were measured by recording the number of bytes in each packet at the probe points. The queue length was increased by the length of the packet specified in

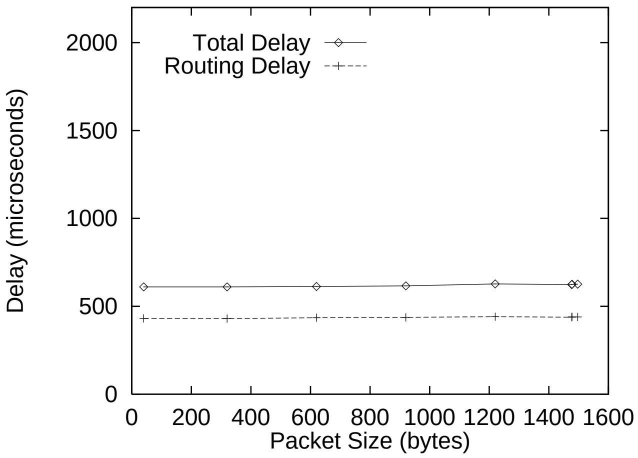

Figure 6.7: Processing delays on the Foreign Agent

the packet header when a packet was enqueued, and decreased by the packet length when a packet was dequeued. The queue lengths observed at each queue exhibited repeating patterns, and are presented in Table 6.2 for each Agent for the minimal input IP packet size (20 bytes in the Home Agent, and 40 bytes in the Foreign Agent), and maximal input IP packet size (1497 bytes in both Agents).

Table 6.2: Queue lengths (in bytes) as monitored on the Agents

Home Agent				**Foreign Agent**			
Minimal Packets		*Maximal Packets*		*Minimal Packets*		*Maximal Packets*	
IP	Interface	IP	Interface	IP	Interface	IP	Interface
0	0	0	0	0	0	0	0
20	40	1497	1492	40	20	1497	1477
0	0	0	0	0	0	0	0
20	40	1497	45	40	20	1497	1477
.	.	.	0	.	.	.	.
.	.	.	1492	.	.	.	.
			0				
			45				
			.				
			.				

The queueing delays at both the IP and interface queues of both Agents were insignificant, and hence queues of packets never built up. The routing subsystem was always able to process packets at a rate faster than the rate at which packets were enqueued in the IP queue. Also, the network interface was always able to transmit outgoing packets at a rate faster than the rate at which packets were enqueued in the interface queue.

6.4.3 Throughput Capacity

The throughput capacity measures have been derived from the delay measures presented in Section 6.4.1 by dividing the number of bits of input data routed through the Agent by the total delay experienced by the packet in the Agent.

Home Agent Throughput Capacity

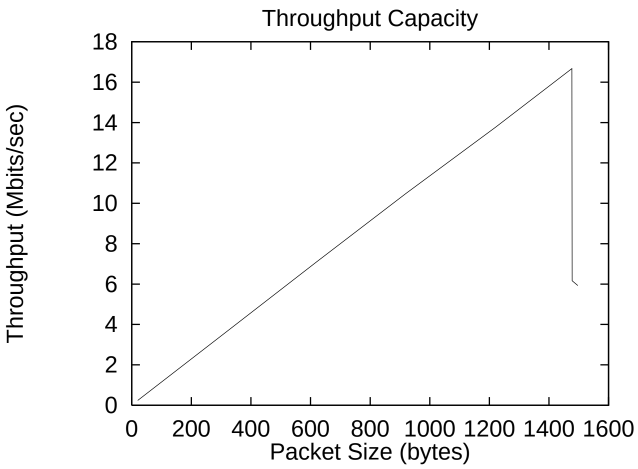

Figure 6.8: Throughput Capacity on the Home Agent

As seen in Figure 6.8, Home Agent throughput capacity increases linearly with increasing packet size, until the MTU of the outgoing interface is reached. When the MTU is reached, fragmentation occurs. This results in copying of data and transmission of two fragments on the outgoing interface, causing a steep drop in the measured throughput capacity at the Home Agent. The maximum throughput capacity is 16.67 Mbits/sec, measured for an incoming packet size of 1477 bytes, the maximum packet size without fragmentation on the outgoing interface. When fragmentation occurs, *i.e.*, for an incoming packet size of 1478 bytes, the observed throughput capacity drops abruptly to 6.16 Mbits/sec.

Foreign Agent Throughput Capacity

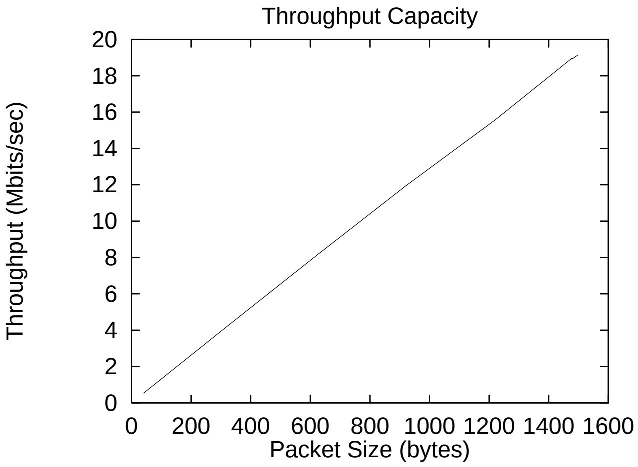

Figure 6.9: Throughput Capacity on the Foreign Agent

As seen in Figure 6.9, Foreign Agent throughput capacity also increases linearly with packet size. Since fragmentation does not occur on the Foreign Agent, no drop in throughput capacity is observed, as in the Home Agent. The maximum throughput capacity observed is 19.13 Mbits/sec, measured for an incoming packet size of 1497 bytes. The maximum throughput capacity is greater than that observed in the Home Agent since the throughput capacity on the Foreign Agent continues to increase beyond the point where fragmentation occurs on the Home Agent.

6.4.4 Analysis of the Measurements

The measurements showed relatively constant delays in all 4 queues that were monitored. The routing delay and total delay in both Agents were also relatively constant, except when packets were fragmented in the Home Agent.

The throughput capacity measurements presented in this section are indicators of the speed at which the Agents can successfully route packets to mobile hosts. They denote the rates at which the software running on the Agents can process packets destined for MNs. They do not reflect the hardware interface card transfers to and from the physical link, which are limited by the characteristics of the Ethernet link, and the transmission rate of the *pumper* process.

The theoretical maximum throughput of the Ethernet to which the Agents are attached is 10 Mbits/sec. In practice, protocols running over Ethernet exhibit substantially lower throughput due to protocol-specific reasons and because Ethernet uses a shared link. In the case of both Agents, the observed throughput capacity of the routing mechanism was found to be greater than the capacity of the Ethernet. The throughput of the incoming packets on the Agents *must* be less than 10 Mbits/s, the theoretical maximum throughput of the Ethernet through which they arrive. The throughput of these packets is limited by factors such as contention for the Ethernet segment that is shared by several hosts and pre-emption of the *pumper* process due to processor scheduling. The fact that the measured throughput capacity of the routing mechanism is greater than that of the Ethernet accounts for the absence of any queue build up on both agents. Packets that are enqueued are immediately dequeued and processed by the routing mechanism before the next incoming packet arrives.

These measures indicate that the design of the Agents is capable of comfortably handling anticipated load imposed for the support of mobile hosts in the context of network technologies that are popular today, such as Ethernet. The measured throughput capacity numbers suggest that the Mobile IP packet processing in this implementation will not be the bottleneck for mobile network users. The real throughput performance limitation will be determined by the physical layer transmission capacity, data copying, interrupt processing and operating system overhead.

6.5 Summary

Simple experimental tests were carried out to demonstrate the functionality of the implemented Mobile IP agents. Three scenarios were demonstrated using the implemented systems, and observed using a network monitor: datagram delivery failure in the absence of Mobile IP, datagram tunneling by the Home Agent, and datagram delivery by the Foreign Agent. The tests served to demonstrate the datagram forwarding capability of the mobile unicast routing mechanism.

The performance of the routing mechanism was measured by instrumenting the kernel with probes, and experiments were coordinated using a framework to schedule measurement runs and to synchronise the execution of the components involved in the experiments.

It was concluded that the overheads of the routing mechanism were within satisfactory limits. When fragmentation occurred on the Home Agent, the maximum delay was found to be 2021 microseconds. When fragmentation did not occur on the Home Agent, the delay was close to the maximum delay of 627 microseconds observed on the Foreign Agent. The routers were found to have peak throughput capacities of 16.67 Mbits/sec on the Home Agents and 19.13 Mbits/sec on the Foreign Agent. However, on the Home Agent, the observed throughput capacity drops abruptly to 6.16 Mbits/sec when fragmentation occurs. The peak throughputs of the Agents are greater than the capacity of an Ethernet, and much greater than the throughput of wireless networks implemented using technologies that are popular today. Even when fragmentation occurs, the throughput capacity was found to be a significant fraction of the theoretical maximum capacity of Ethernet. Hence, the Agents, implemented as described in this thesis, will not constitute a bottleneck in the network infrastructure for the support of mobile hosts.

Chapter 7

Multicast for Mobile Hosts

The provision of services to mobile users is rapidly emerging as a fundamental requirement of network design. Several mobile computing services could be effectively provided over an underlying multicast facility. Among these might be the dissemination of information such as weather reports, travel information and stock market reports. In addition, applications like mail, multi-person communication, service location and distributed systems functions like cache consistency could also make use of a general purpose multicast facility.

In a mobile environment, the network must not only manage multicast group membership and establish the necessary routes, but must also contend with the fact that the established routes are themselves transient in nature. The fact that the network must deal not only with dynamic group membership but also with the dynamic location characteristics of mobile nodes makes multicast in a mobile environment a challenging problem.

7.1 A Previous Approach

An approach to multicasting for mobile hosts was first proposed by Acharya and Badrinath [AB93]. Their scheme assumes static multicast groups, i.e., it assumes that the membership of any group does not change during the group's lifetime. The multicast problem is approached as being distinct from, and independent of, any particular solution for unicast routing. Continuous coverage for mobile hosts is facilitated by augmenting the static network with *mobile support stations* (*MSS*s, similar to, but not the same as the *agents* in Mobile IP) that are each capable of direct communication with mobile hosts within a limited geographic area. All MSSs are assumed to be capable of providing multicast services for visiting mobile hosts.

The Acharya and Badrinath scheme manages location information on a *per-group* basis. A *host view* is defined as the set of all MSSs that have members of a particular multicast group. A *host view membership protocol, HVMP*, is executed within the static network to maintain and update a group's host view. Information on the members of a host view is replicated on all its members. Changes to this replicated information is achieved upon every change to the host view by synchronisation via a centralised coordinator, followed by notification to all the MSSs in the host view. This information is then utilised to deliver multicast messages to the group of MNs corresponding to the host view. When a host wishes to send a multicast datagram, the datagram is sent to a directly reachable MSS (the initiator), which then forwards the datagram to all the MSSs in the host view of the group to which the datagram is addressed.

An implementation of the scheme would use direct physical links or tunnels between the initiator and each of the other MSSs in the host view in order to perform datagram delivery. The scheme implies that even if tunnels to multiple MSSs in the host view share common physical links, multiple datagrams will still be transmitted on those links. The scheme relies on a *hand-off* procedure very similar to that used in the Columbia proposal for unicast routing [IDM91], to transfer a MN's state from its previous cell to its current cell after a move. The scheme assumes that all the system components (hosts, MSSs and communication links) are reliable and that MSSs have sufficient memory to *buffer* multicast messages as needed in order to provide *exactly once* delivery semantics. They focus solely on host mobility and consider multicast messages that are addressed only to mobile hosts. If a host is not a member of a group, it can multicast only if it is local to a MSS that belongs to the host view.

7.2 A New Approach

This research considers the problem of multicast to dynamic groups in an internetwork with mobile hosts. Adapting a static host multicast mechanism such as that provided by IGMP and DVMRP in the TCP/IP suite to support mobile nodes as well would seem non-trivial. However, a deployed, proven multicast mechanism that already exists for static hosts on the Internet is potentially of great utility if it can be leveraged to provide multicast services for mobile hosts as well.

An intuitive scheme for providing multicast as a service for mobile nodes is proposed, which makes use of the foundation provided by a Mobile IP implementation and IP multicast. The goal of the scheme is to provide multicast capability to mobile hosts transparently in a manner similar in function to that available on the Internet today. The following assumptions have been made:

- The disruption of multicast service due to movement of a host from one network to another must be minimised.
- The service to be provided is the *unreliable, best effort, connectionless* delivery of multicast datagrams, i.e., datagrams may be lost, duplicated, delayed or delivered out of order. Higher level protocols are responsible for handling such conditions. This feature, a basic design tenet of the network layer in the Internet, has contributed to its robustness and is largely responsible for its success [Com91].
- Dynamic group management is a necessary feature of multicast.
- Both static and mobile hosts can be members of multicast groups.
- A mobile host that wishes to receive multicast datagrams is capable of independently receiving them on its home network without any additional support,

using existing multicast techniques. In the proposed architecture, it is assumed that a multicast router is co-resident with the HA, and is denoted as *mHA*.

7.2.1 Handling Multicast Source Mobility

One approach to solving the problem associated with a mobile *source* of multicast datagrams is to make the multicast network react to changes in the location of the source of multicast datagrams. This would potentially involve changing the algorithm currently used for IP multicast, possibly using some mechanism to invalidate old routes upon host movement. Such an approach seems prohibitively expensive in terms of the cost of repeatedly modifying incorrect routes, and the repeated tree set-up cost. The reuse of the multicast tree once set up is desirable. To ensure interoperability with DVMRP, the approach proposed guarantees that the multicast tree is always rooted at the home network of the MN. When the MN wishes to send multicast datagrams,

- if it is on its home network, it uses link-level multicast to send the datagram. The mHA propagates the multicast downstream normally.

- if it is on a foreign network, it uses a tunnel to deliver the datagram to its mHA. The mHA then propagates the multicast datagram downstream via all virtual interfaces, including the physical network interface corresponding to the MNs home network. The recipients of the multicast are unaware that the source is mobile.

Higher level protocols that implement retransmission mechanisms such as negative acknowledgements (NAKs) unicast these datagrams to the MN. Mobile IP software on the mHA provides for the delivery of these datagrams to the MN. The result is transparent operation in the face of multicast source mobility.

7.2.2 Handling Multicast Destination Mobility

Future public wireless networks such as those used for large scale information dissemination services will likely be served by one or a few centralised sources of information e.g., stock market reports originating from a few stock exchanges, or weather reports from a regional weather bureau. This information would be received by a very large number of MNs via a large number of access points. Thus in order to support mobile *destinations* of multicast datagrams the network requires

- a few access points to a general multicast infrastructure (the mHAs), and
- several access points to permit the dissemination of messages to the MNs (the FAs).

The underlying philosophy of the proposed approach is that HAs are static and also know the location of the MNs they serve. Thus it is logically feasible for a multicast HA, a mHA, to forward multicast traffic to the MN through the MIP tunnel via the FA. This occurs transparently to software on the MN. By not maintaining group information at all, passing this burden instead onto a tried and tested protocol, the resulting scheme is simple. Each FA needs to worry only about the MNs it is currently serving rather than all the agents serving all the members of all the groups represented on its local network. This information is mostly already known to the FA through the registration procedures used in Mobile IP.

As shown in Figure 7.1, the HA receives multicast datagrams via the IP multicast distribution tree set up using DVMRP. Since the HA may be serving MNs at several FAs that wish to receive datagrams addressed to multicast address X, it forwards a copy into each corresponding Mobile IP tunnel.

This scheme is complicated by the *tunnel convergence problem* resulting from the fact that multiple MIP tunnels (from different HAs) can terminate at a given FA. Thus one copy of every multicast packet would be forwarded to the FA by each HA

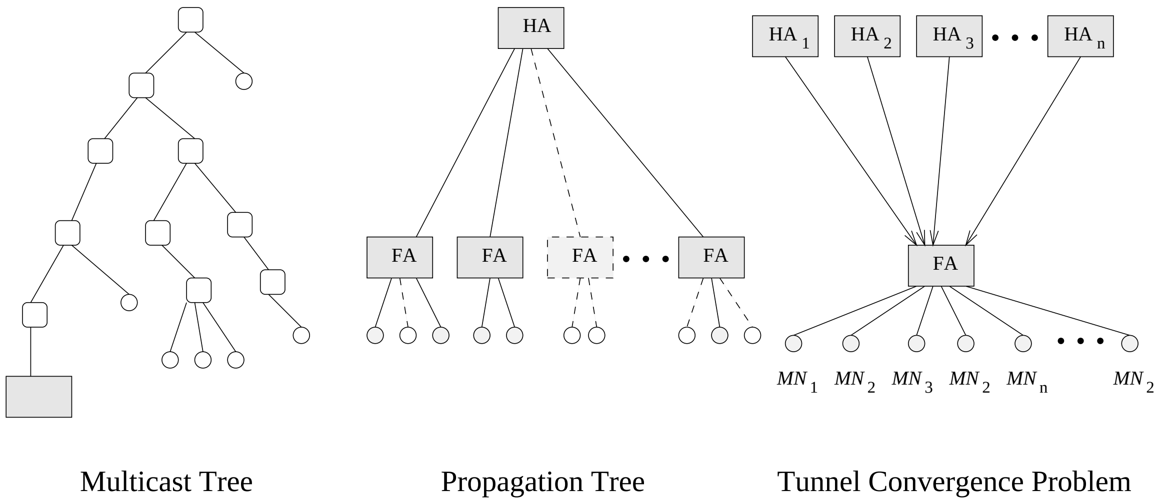

Figure 7.1: Multicast packet paths under proposed scheme

that is serving interested MNs. Since the FA would locally deliver every multicast datagram forwarded to it, the problem of duplicate multicast packet delivery to the MNs must be dealt with. While duplicate datagrams do not constitute a violation of the service assumptions, they constitute an additional load on a possibly low-bandwidth network. To solve this, a simple selection based on maximum number of MNs served is performed by the FA to appoint one HA as the *designated multicast service provider, DMSP*, for a given multicast group. The DMSP forwards only one multicast datagram into the tunnel even if it serves multiple MNs at the foreign network. Thus the scheme provides *at-most-once* delivery of multicast datagrams which is identical to the semantics of IP multicast. The at-most-once delivery semantics may not hold true if multiple FAs are located on the same network, but this is not a problem since duplicate datagram delivery is permitted.

7.2.3 Service Mechanism

All multicast transmissions by the MN are tunneled directly to the mHA, instead of the use of hardware multicast. Any recipients on the foreign network receive the datagrams through the multicast tree rather than directly via link-level multicast by the MN. The steps involved in providing multicast service for mobile recipients are:

Service Request: The MN wishing to receive multicast datagrams informs the FA that it wishes to join a specific multicast group. Ordinarily, to join a group, an Internet host sends an IGMP *membership report* in reply to the multicast router's *membership query*. This is addressed to the multicast address X, corresponding to the group being joined, and multicast routers listen promiscuously to all LAN multicast traffic. The IGMP software on the MN is modified to perform a check to determine whether the MN is on its home network. If the MN is not on its home network, the membership report is addressed explicitly to the FA instead of being multicast on the foreign network.

Service Processing: If the FA supports multicast service, it propagates the request to the mHA. Also, if the MN is the first mobile node on the FA's network to request service for address X, the FA also informs the mHA of the mHA's status as the DMSP, for address X. The mHA does not prune itself from the multicast distribution tree, and receives all multicast datagrams addressed to X.

Service Execution: The multicast service is provided when the DMSP forwards all datagrams received corresponding to multicast address X into the MIP tunnel. When a MN moves from one network to another, no notice is given to the previous FA, whereas the mHA receives explicit information regarding the MN's new location when the MN attempts to re-register through the new FA. Rather, Mobile IP [Per95] specifies that the FA eventually times out and discards the MN's registered status when it fails to receive a re-registration request in a specified period. This implies that if the DMSP has only one MN at a foreign network interested in receiving multicast datagrams sent to address X, and that MN moves to another location, the tunnel to the old FA is discarded when a registration request by the MN is received through another FA. This leaves the other group members at the previous foreign network without a DMSP.

Service Handoff: It is imperative that the FA be informed about the inability of a mHA to continue as the DMSP. A mechanism similar to that used for large-scale distributed file system consistency [HKM+88] is used. For the first group member at a foreign network, the corresponding mHA is the DMSP. As part of the service setup procedure, the mHA issues a *callback promise* to the FA. It retains this status until it relinquishes it when the last MN it serves at the foreign network that is also a group member moves elsewhere. At this point of time, the DMSP informs the FA that it is no longer willing to provide the multicast service for address X, a procedure called a *callback break*. This triggers the selection of a new DMSP for address X by the FA if any MNs remain that are still members of the group corresponding to address X. This callback mechanism ensures that the break in service for MNs is minimised due to the use of a handoff procedure. Since the next DMSP is receiving the multicast datagrams, it can simply enable the forwarding of these datagrams into the tunnel.

7.3 Limitations of the Proposed Approach

This section presents a critical evaluation of the scheme proposed in the previous section and defines additional requirements for IP multicast in a mobile environment.

The limitations of the scheme proposed in Section 7.2 are:

- Packets that are sent and received by MNs must always traverse the home network, making routing non-optimal.
- The DMSP optimisation suffers from a scoping problem: IP multicast packets are scoped to limit their propagation using a TTL (time-to-live) field. Multicast packets that are transmitted on Home Networks other than the one the DMSP is on, may not make it to the Foreign Network.
- Multiple unicasts are used by the HA to tunnel multicast packets to FAs of MNs that are group members. However, multiple unicasts are often used by

the multicast routers on the MBONE to tunnel multicast packets to multicast-capable islands in the Internet.

Non-optimal Routing Non-optimality may be defined as the degree to which the route taken by a packet deviates from the path IP would have routed the packet to the destination if the sender knew the location of the MN, and could have it routed to the destination by the shortest possible path. Non-optimality is not seen as a major drawback because it is unavoidable in the case of the sending MN, and it is widely held that several if not all services used on a MN will be accessed from the Home Network, making routing optimal. Further, non-optimality is defined in the context of IP routes and not physical paths, and IP itself can establish non-optimal paths (when network load and other parameters are considered). Finally, optimal solutions are generally more complex, and the complexity may not be justified.

Lack of Support for Small-Scope Groups A *small-scope group* may be defined as a group such that multicasts addressed to that group are constrained to remain within a relatively small region of an internetwork. The problem of multicast support for small-scope groups was not considered. This is because the precise nature of the support that needs to be provided for mobile hosts is still very much in the process of being defined. For example, it was only recently, in April 1995, that the IETF Mobile IP Working Group decided that LAN broadcasts must be forwarded to roaming MNs. Newer TCP/IP-based protocols use IP multicast in preference to IP broadcast whenever possible on a LAN to reduce the load on hosts that are not interested in the transmission. Since broadcasts are to be tunneled, it seems logical that network support also be provided for small scope multicast groups, such as that used in preference to the IP broadcast address.

Possible Use of Multiple Unicasts The multiple unicast problem arises if the HA is serving more than one mobile group member for a given group, and these MNs are at different Foreign Networks. The HA must therefore make multiple copies of the multicast packet, and then, depending on whether it is serving as the DMSP for a particular Foreign Net, tunnel the packet to the corresponding FAs. As the performance experiments for fragmented packets described in Chapter 6 showed, making multiple copies and transmissions per incoming packet has a high associated cost, one that reduces the number of hosts that a HA can serve, and, as a consequence, the scalability of the system. In the performance experiments fragmentation results in a copy of only a subset of the total number of bytes in the packet and two transmissions, which reduces the throughput markedly. In the case of multicast and broadcast packets, a much larger number of copies of whole packets and multiple transmissions of these copies may be needed, and the resulting impact on performance would be substantial. It is indeed conceivable to improve the scheme further by using multicast itself (i.e., multicast Mobile IP tunnels) to propagate the multicast packet to the MNs using only a single transmission by the HA.

7.4 Summary

This chapter described a previous approach for providing multicast to mobile hosts and proposed a new approach. The previous approach seeks to provide *exactly-once* delivery of multicast datagrams, but has several limitations that make it unsuitable for practical use. It assumes static groups and manages location information on a per-group basis using a *host-view membership protocol, HVMP*, executed on the static network.

The proposed multicast scheme is based on an expansion of the virtual network established for IP Multicast by using Mobile IP tunnels to constitute the "last mile" for delivery of multicast datagrams. The scheme has several features that make it practical as a solution for mobile hosts on TCP/IP networks. In particular, it provides the ability to support dynamic groups and provide minimal break in service as a result of host movement.

The proposed approach has limitations, however, including non-optimal packet routing, lack of support for small-scope groups, and the possibility of multiple unicasts being used by the Home Agents to propagate multicast datagrams to more than one Foreign Network.

Chapter 8

Summary and Conclusions

This chapter presents a summary of the work done in this thesis, the conclusions drawn from this work, and the contributions made by it. Future research directions are also suggested.

8.1 Summary

The thesis consists of four major parts:

- a review of proposals for providing mobility support in TCP/IP networks
- an implementation of the Mobile IP protocol in a 4.3 BSD Unix environment
- an investigation of the cost of mobility support through experimental measurements of the Mobile IP implementation
- a proposal for providing multicast support in a mobile environment, on top of Mobile IP

Investigation of Mobility Support in TCP/IP Networks: A survey of proposed schemes to provide mobility support at the network layer in TCP/IP networks was carried out. Several protocols, including Virtual Networks, Virtual Internet Protocol, Columbia Mobile IP, IETF Mobile IP, and others, were reviewed. One of these, the IETF Mobile IP protocol, is likely to gain widespread acceptance and was thus selected for implementation and evaluation.

Mobile IP Implementation: The development of a prototype implementation of Mobile IP was carried out in the following stages:

1. Problem analysis and architectural design
2. Design, implementation and test of the Unix kernel components involved
3. Design and implementation of the user-space components
4. Integration and test of the system

Performance Evaluation of Prototype by Measurements: The performance of the routing mechanism was measured by kernel instrumentation using probes, and using a framework developed for the scheduling of measurement runs and the synchronisation of the components involved in the experiments.

Multicast for Mobile Hosts: The principal goal was the design of an original scheme to provide multicast services for mobile hosts. This was done in the following stages:

1. Survey and evaluation of existing approaches.
2. Development of a proposal for handling multicast in the context of the unicast Mobile IP facility [CBW95].

8.2 Conclusions

The following conclusions could be drawn from this research:

- The design choices made in implementing mobile routing support were effective.
- The fragmentation overhead due to tunneling which is used in Mobile IP is significant.

- The observed throughput capacity measures of the Agents (a peak of 16.6 and 6.6 Mbps before and after fragmentation in the Home Agent, and 19.13 Mbps in the Foreign Agent) compares favourably with underlying network technologies.
- Multicast for mobile hosts can be provided by suitable interoperation of Mobile IP and static IP multicast.

8.3 Relevance and Contributions

This work is relevant because it has immediate applicability to the burgeoning market for laptop computers, wireless networks and PDAs. The following contributions were made:

- it has immediate applicability to the burgeoning market for laptop computers, wireless networks and PDAs
- it presents a unique design for the integration of routing support for mobile hosts in a TCP/IP internetwork
- it provides a working implementation of Mobile IP, which is to become an Internet standard protocol for the support of mobile hosts
- it provides basic measurement data on a real-world experimental prototype of an approach to providing network support for mobile hosts
- it addresses the issue of multicast to mobile hosts, which has several practical applications and is of immediate interest to the Internet community

Two specific contributions relating to the design and specification of the Mobile IP protocol itself arise from the practical experience gained in the design and implementation process:

- Mobile IP solves the problem of routing of packets *to* mobile nodes. The protocol draft did not state anything about routing of packets sent by the mobile node upon each movement to a new network. Practical experience showed that upon each movement, the mobile node would carry stale entries in its IP routing tables and ARP cache with reference to directly connected nodes on its previous network. This would cause misrouting of packets sent by the mobile node upon each movement until timeouts caused stale entries to be individually deleted. It was suggested to the Mobile IP Working Group that the mobile node flush its routing table upon each movement.

- The packet interception mechanism to perform Mobile IP required that the mobile node issue a Gratuitous ARP upon its return to its home network. Implementation experience showed that while theoretically feasible, this was very cumbersome to perform since it required the use of very low level interface-level access. Without this requirement, an mobile node could be set up with only user-level software.

8.4 Directions for Future Research

Future research issues cover four broad areas:

- **Scaling issues:** These concern how the network infrastructure in general, and the routing mechanisms in particular, scale with a very large number of mobile hosts, rapid host movement, and an increase in the volume of traffic. In the mobile computing community, it is widely believed that mobile computing devices will become ubiquitous and widely used for both interactive and embedded applications. The ability of unicast and multicast routing mechanisms to scale to large numbers of hosts is of great interest. Rapid host movement between networks for reasons such as signal strength fluctuations in overlapping wireless cells will also require routing schemes to adapt accordingly. Finally, as the volume of traffic increases as a result of increased usage, the network infrastructure must scale gracefully to accommodate the increased load. These and other issues will become more apparent and clearly defined as more experience is gained with real implementations.

- **Wireless issues:** This research focussed solely on Network layer issues. The use of wireless networks for internetwork access does not affect routing in itself, since all interactions between the mobility Agents would take place on a wired backbone. It does, however, affect applications since these networks exhibit different physical characteristics such as high error rates. Recent research has shown that traditional network mechanisms such as those employed in the Transport layer are inappropriate [BBIM93], and new schemes must be explored. There is considerable work to be done in this area, especially since any solution must not require modifications to existing Transport layer software on static hosts.

- **Service issues:** Mobile environments are likely to see new paradigms of service interaction. In a mobile environment, *client-server* architectures face problems such as service discovery, and incompatibility between protocols used by a mobile client and a fixed server. This has created a need for the use of standard interfaces and service transparency using architectures such as CORBA [Gro93] and OSF DCE [OSF92]. Several open research issues remain related to interoperation and protocols for service interaction.

- **High-bandwidth and QOS issues:** Wired networks will see a slow shift towards high bandwidth technologies such as Asynchronous Transfer Mode (ATM). At the very least, it will be essential for mobile networking protocols to interoperate with the new technologies on the wired network. More ambitiously, as wireless networks themselves achieve higher throughputs, it becomes possible to provide applications with quality of service guarantees for applications such as multimedia communication.

Bibliography

[AB93] A. Acharya and B. R. Badrinath. Delivering multicast messages in networks with mobile hosts. In *Proc. 13th International Conference on Distributed Computing Systems*, pages 292–299, Pittsburgh, PA, May 1993.

[BBIM93] B. R. Badrinath, A. Baker, T. Imielinski, and R. Marantz. Handling mobile clients: A case for indirect interaction. In *Proc. 4th Workshop on Workstation Operating Systems*, pages 91–97, Napa, CA, October 1993.

[BFC93] T. Ballardie, P. Francis, and J. Crowcroft. Core Based Trees (CBT): An architecture for scalable inter-domain multicast routing. In *Proc. ACM SIGCOMM '94 Conference: Communications Architectures and Protocols*, pages 86–95, Ithaca, N.Y., USA, September 1993.

[BLCGP92] T.J. Berners-Lee, R. Cailliau, J-F. Groff, and B. Pollermann. World-Wide Web: The information universe. *Electronic Networking: Research, Applications and Policy*, 2(1):52–58, Spring 1992.

[CBP93] K. C. Claffy, H. W. Braun, and G. C. Polyzos. Traffic characteristics of the T1 NSFNET backbone. In *Proc. IEEE INFOCOMM '93*, volume 2, pages 885–892, San Francisco, CA, 1993.

[CBW95] V. Chikarmane, R. Bunt, and C. Williamson. Mobile-IP based multicast as a service for mobile hosts. In *Proc. Second International Workshop on Services in Distributed and Networked Environments (SDNE'95)*, pages 11–18, Whistler, B.C., Canada, June 1995.

[Chi94] V. Chikarmane. Disconnected operation in a distributed system. In *Proc. Annual Graduate Student Symposium*, pages 23–32, University of Saskatchewan, Saskatoon, Canada, April 1994. Computer Science Department.

[Com91] D. E. Comer. *Internetworking with TCP/IP*, volume 1. Prentice-Hall Inc., Englewood Cliffs, NJ, second edition, 1991.

[DC90] S. Deering and D. Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

[Dee89] S. Deering, editor. *Host Extensions for IP Multicasting*, RFC 1112, August 1989.

[Dee91] S. Deering, editor. *ICMP Router Discovery Messages*, RFC 1256. Xerox PARC, September 1991.

[DEF+94] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C-G. Liu, and L. Wei. An architecture for wide-area multicast routing. In *Proc. ACM SIGCOMM '94 Conference: Communications Architectures and Protocols*, pages 126–135, London, England, U.K., August 1994.

[DZ83] J. D. Day and H. Zimmermann. The OSI reference model. *Proc. IEEE*, 71:1334–1340, December 1983.

[Eri94] H. Eriksson. MBONE: The multicast backbone. *Communications of the ACM*, 37(8):54–60, April 1994.

[EW94] O. Etzioni and D. Weld. A softbot-based interface to the Internet. *Communications of the ACM*, 37(7):72–76, July 1994.

[FZ94] G. H. Forman and J. Zahorjan. The challenges of mobile computing. *IEEE Computer*, 27(4):38–47, April 1994.

[Gro93] Object Management Group. Common Object Request Broker Architecture (CORBA) 1.2 specification draft. December 1993.

[HH93] L. Huston and P. Honeyman. Disconnected operation for AFS. In *Proc. 1993 USENIX Symposium on Mobile and Location-Independent Computing*, pages 1–10, Cambridge, MA, August 1993.

[HKM+88] J. H. Howard, M. L. Kazar, S. G. Menees, D. A. Nichols, M. Satyanarayanan, R. N. Sidebotham, and M. J. West. Scale and performance in a distributed file system. *ACM Transactions on Computer Systems*, 6(1):51–81, October 1988.

[HPGP92] J. S. Heidemann, T. W. Page, R. G. Guy, and G. S. Popek. Primarily disconnected operation: Experiences with Ficus. In *Proc. Second Workshop on Management of Replicated Data.* , November 1992.

[IB94] T. Imielinski and B. R. Badrinath. Mobile wireless computing. *Communications of the ACM*, 37(10):18–28, October 1994.

[IDM91] J. Ioannidis, D. Duchamp, and G. Q. Maguire. IP-based protocols for mobile internetworking. In *Proc. ACM SIGCOMM '91 Conference: Communications Architectures and Protocols*, pages 235–245, Zurich, Switzerland, September 1991.

[IM93] J. Ioannidis and G. Q. Maguire. The design and implementation of a mobile networking architecture. In *Proc. Winter USENIX Conference*, pages 489–500, San Diego, CA, January 1993.

[Int87] SRI International. Network reconstitution protocol. Technical Report RADC-TR-87-38, Rome Air Development Center, June 1987.

[Ioa93] J. Ioannidis. *Protocols for Mobile Internetworking.* PhD thesis, Columbia University, 1993.

[JMP94] D. B. Johnson, A. Myles, and C. Perkins. The internet mobile host protocol (IMHP). Internet Draft, February 1994.

[Joh94] D. B. Johnson. Scalable and robust internetwork routing for mobile hosts. In *Proc. 14th International Conference on Distributed Computing Systems*, pages 124–133, June 1994.

[KS92] J. J. Kistler and M. Satyanarayanan. Disconnected operation in the coda file system. *ACM Transactions on Computer Systems*, 10(1):3–23, February 1992.

[LMKQ89] S. J. Leffler, M. K. McKusick, M. J. Karels, and J. S. Quarterman. *The Design and Implementation of the 4.3 BSD UNIX Operating System.* Addison-Wesley Publishing Co. Inc., 1989.

[Mal94] G. Malkin, editor. *RIP Version 2 Carrying Additional Information*, RFC 1723. Xylogics, Inc., November 1994.

[Moy94a] J. Moy. Multicast routing extensions for OSPF. *Communications of the ACM*, 37(8):61–66, August 1994.

[Moy94b] J. Moy, editor. *OSPF version 2*, RFC 1583. Proteon, Inc., March 1994.

[MP85] J. Mogul and J. Postel, editors. *Internet Standard Subnetting Procedure*, RFC 950. USC/Information Sciences Institute, August 1985.

[MP93] A. Myles and C. Perkins. Mobile IP extensions (MIP). Internet Draft, October 1993.

[MS93] A. Myles and D. Skellern. Comparison of mobile host protocols for IP. *Proc. Internetworking Research and Experience*, 4:175–194, December 1993.

[Neu83] J. W. Neumann. OSI transport and session layers: service and protocols. In *Proc. IEEE*, pages 115–123, San Diego, April 1983.

[OSF92] Inc. Open Software Foundation. The OSF Distributed Computing Environment. Technical Report OSF-DCE-PD-1090-4, 1992.

[Per95] C. Perkins. IP mobility support. Internet Draft, Mobile IP Working Group, IETF, January 1995.

[Plu82] D. C. Plummer, editor. *An Ethernet Address Resolution Protocol*, RFC 826, November 1982.

[Pos81] J. B. Postel, editor. *Internet Protocol*, RFC 791, Menlo Park, CA, September 1981. NIC, SRI International.

[PP93] C. Papadapoulos and G. M. Parulkar. Experimental evaluation of SUNOS IPC and TCP/IP protocol implementation. *IEEE/ACM Transactions on Networking*, 1(2):199–216, April 1993.

[PR92] C. Perkins and Y. Rekhter. Short-cut routing for mobile hosts. Internet Draft, T.J. Watson Research Center, IBM, July 1992.

[RIOP94] F. Reichert, Y. Ismailov, T. Ohsawa, and P. Pruthi. The walkstation project - a flexible platform for experiments in mobile multimedia environments. In *Proc. IFIP Congress 94*, Hamburg, Germany, Aug-Sept 1994.

[Sim94] W. A. Simpson. IP mobility support. Internet Draft, Mobile IP Working Group, IETF, March 1994.

[SP80] C. Sunshine and J. Postel, editors. *Addressing Mobile Hosts in the ARPA Internet Environment*, IEN 135, March 1980.

[Ste90] W. R. Stevens. *UNIX Network Programming*. Prentice-Hall Inc., Englewood Cliffs, NJ, 1990.

[Tan88] A. S. Tanenbaum. *Computer Networks*. Prentice-Hall, Inc., Englewood Cliffs, N.J., second edition, 1988.

[TCT92] F. Teraoka, K. Claffy, and M. Tokoro. Design, implementation, and evaluation of virtual internet protocol. In *Proc. 12th International Conference on Distributed Computing Systems*, pages 170–177, June 1992.

[TT93] F. Teraoka and M. Tokoro. Host migration transparency in IP networks: the VIP approach. *ACM Computer Communication Review*, 23(1):46–65, 1993.

[TYT91] F. Teraoka, Y. Yokote, and M. Tokoro. A network architecture providing host migration transparency. In *Proc. ACM SIGCOMM 91*, pages 209–220, Zurich, Sitzerland, September 1991.

[UTSM93] K. Uehara, F. Teraoka, H. Sunahara, and J. Murai. Enhancement of VIP and its evaluation. In *Proc. International Networking Conference (INET)*, pages BEA–1 – BEA–10, San Francisco, California, August 1993. Internet Society.

[WOM92] H. Wada, T. Ohnishi, and B. Marsh. Packet forwarding in mobile hosts. Technical Report MITL-TR-39-92, Matsushita Information Technology, December 1992.

[WPD88] D. Waitzman, C. Partridge, and S. Deering, editors. *Distance Vector Multicast Routing Protocol*, RFC 1075. BBN STC, Stanford University, November 1988.

Appendix A

Mobile IP Router Pseudocode

The design has integrated both HA and FA functionality into the same router. The principal modules involved in mobile routing previously shown in Figure 4.6, have been reproduced in Figure A.1 for cross-reference with the pseudocode of the basic Mobile IP kernel subsystem presented in this appendix. Calls to routines not described in this section are to existing kernel routines (eg. *mbuf* manipulation routines). The implementation has merged and separated the modules described in the pseudocode into functions as convenient.

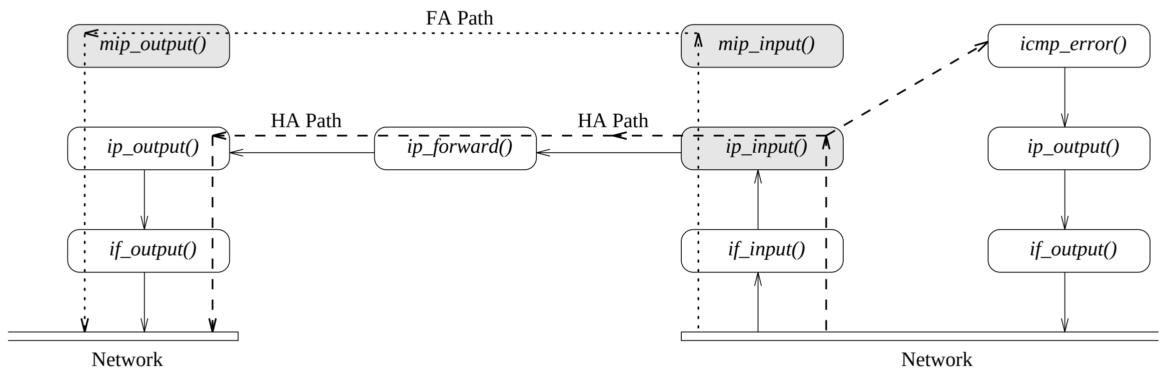

Figure A.1: Paths taken by packets through the routing mechanism

Processing is performed specific to each incoming packet. Depending on the role the router is performing with reference to an incoming packet, the packet can take either

- the *HA path* where Home Agent processing is performed. This is described in section A.1.

- the *FA path* where Foreign Agent processing is performed. This is described in section A.2.

A.1 Home Agent Processing

This is shown by the dashed line in Figure A.1.

A.1.1 The IP Input Module

IP's input module is executed upon a software interrupt scheduled by the network device driver. This module[1] must contain the functionality to deal with both tunneled packets (when the router is serving as some MN's FA, corresponding to the dotted FA path in Figure A.1), and ordinary packets from some CH to a MN (when the router is serving as that MN's HA corresponding to the dashed HA path in Figure A.1), as described in Section 2.3.

Packets entering the IP software are looked up in the Agent Table for a match against the destination of the packet. A successful lookup indicates that the packet is meant for an MN. If the router is serving as a HA for that MN, the packet must be an ordinary packet and is tunneled. If the router is not the HA, then the packet is dropped to eliminate the possibility of a routing loop occurring.

A packet addressed to the router itself could potentially be an encapsulated IP packet meant for a MN for whom the router is serving as a FA. These packets are demultiplexed by switching directly to `mip_input()` as shown in Figure 4.3(b), using the `inetsw[]` switch table as shown in Figure 4.3(c).

```
ip_input()
{
 ...
```

[1] called ipintr() in UNIX

```
 ...
 lookup Agent Table by calling find_at();


 if (lookup succeeded, ie. the pkt is for an MN) {
    if (this machine is the HA of the MN)
       call mip_tunnel() to tunnel the pkt to the FA;
    else
       drop pkt;
 }
 ...
 if (the pkt is NOT for this machine)
    call ip_forward() to transmit the possibly encap pkt ;
 ...
 if (the pkt is for this machine) {
    reassemble datagram;
    pass datagram up to the protocol layer (MIP/UDP/TCP etc),
       ie. switch to pr_input() of the encap pkt, possibly mip_input()
       (*inetsw[ip_protox[ip_proto]].pr_input)(m, ifp);
 }


 the pkt is bad, so drop it;
}
```

A.1.2 The MIP Tunneling Module

This module determines how to encapsulate the packet-to-be-tunneled by calling `mip_softstate()` to determine whether or not the MIP-encapsulated packet will fragment in the tunnel as shown in Figure 4.5, and then calls the appropriate routine to perform the encapsulation. The implementation presently uses a default IP-in-IP encapsulation. The call to `ip_forward()` releases the encapsulated packet into the tunnel, shown as the left-fork of the dashed line in Figure A.1.

```
mip_tunnel(ip, ifp, ro)
{
 if (rtentry for MN does not have a pointer to a TCB in its rt_tcb field)
    return(TCB_DOESNTEXIST);
 else
    tun = TCB pointed to in the rtentry for the MN;
 fragstatus = mip_softstate(ip, tun);
 switch(fragstatus) {
    case WILLFRAG: mip_encapin_ip(ip);
                   break;
    case WONTFRAG: mip_encapin_mip(ip);
                   break;
 }
}
```

A.1.3 Encapsulation using the IP header

This routine performs IP-in-IP encapsulation on the packet-to-be-tunneled as shown in Figure 4.4. The resetting of the DF (Dont Fragment) flag is important to allow efficient recognition of this packet as IP-in-IP by `mip_input()` at the FA which will need to decapsulate it.

```
mip_encapin_ip(ip, ro)
{
 call m_get() to acquire a new mbuf;
 call mtod() to access a new IP header in the mbuf;
 initialize the fields of the new IP header
   ie. set ip_ttl and ip_tos from inner IP header;
           ip_vers = IPVERSION and ip_hlen = IP_MINHEADER;
```

```
          ip_len = innerIPdatagram's ip_len + IP_MINHEADER;
          ip_proto = IPPROTO_MIP;
          ip_dst = rt_gateway field of rtentry for MH;
          DF flag in ip_flags = 0 to allow fragmentation if needed;
 prepend the new IP header before the old IP header;
}
```

A.1.4 Encapsulation using the MIP header

This encapsulation format has been left unimplemented, but pseudocode has been included for completeness. This routine performs MIP encapsulation on the packet-to-be-tunneled as shown in Figure 4.4. The setting of the DF flag is important to allow efficient recognition of this packet as being MIP encapsulated by `mip_input` at the FA which will need to decapsulate it.

```
mip_encapin_mip(ip, ro)
{
 call m_get() to acquire a new mbuf;
 call mtod() to access a new MIP header in the mbuf;
 initialize the fields of the new MIP header
   ie. set mip_dst = ip_dst of IP header;
          mip_src = ip_src of IP header;
          mip_proto = ip_proto of IP header;
          mip_S = 1 to indicate source addr is present;
          mip_reserved = 0;
          mip_cksum = value returned by in_cksum();

 modify selected fields of the IP header
   ie. set ip_proto = IPPROTO_MIP;
          ip_dst = rt_gateway field of rtentry for MH;
          ip_len += MIP_MAXHEADER;
```

```
          DF flag in ip_flags = 1 to NOT allow fragmentation;
 insert MIP header between IP header and IP payload using m_pullup() and m_adj();
}
```

A.1.5 The MIP Control Input Module

This module receives ICMP error messages from `icmp_input()`, and performs basic updates to the tunnel state by comparing the IP and MIP header fields of the packet from CH to MN that caused the error. This module does not lie on the HA or FA paths shown in Figure A.1. It was concluded that in their present form, the returned header formats do **not** carry information to allow deduction of the TTL of the tunnel. This is because the time-to-live (`ip_ttl`) field in the IP header is decremented at each intermediate router and the enclosed IP packet header will always have `ip_ttl` as zero (which is what in fact, causes the 'time exceeded in transit' ICMP error message to be returned in the first place). Hence it is proposed that the `ip_ttl` field of the packet being tunneled be stored in the second (presently undefined) byte of the MIP encapsulation header, which will always be returned in every ICMP message. Note that only packets encapsulated in MIP headers can be handled this way. Based on this tunnel state mechanism, more sophisticated functionality incorporating additional ICMP types can be developed.

```
mip_cltinput()
{
 search thru list of TCBs, looking for a match;
 switch(icmp_type) {
   case ICMP_UNREACH: set TCB's TUF_UNREACH flag = true;
                      if ((icmp_code == NEEDFRAG) &&
                          (IP pkt's ip_mtu < tu_mtu))
                         set tu_mtu = ip_len;
                      break;
   case ICMP_TIMXCEED: if ((icmp_code == INTRANS) &&
```

```
                         (mip_ttl < tu_ttl))
                        tu_ttl = mip_ttl;
                    break;
 }
}
```

A.1.6 ICMP Processing

This routine compares important attributes of the packet-to-be-tunneled with the current state of the tunnel as shown in Figure 4.5, and fakes ICMP error messages to the CH (shown as the right-fork of the dashed line in Figure A.1) in anticipation of error messages from the tunnel interior that cannot be returned to the CH if and when they do eventually arrive.

```
boolean
mip_softstate(ip, tun)
{
 if (TCBF_UNREACHABLE flag is set)
    if (TCBF_FRAG flag is set) {
       if (IP_pkt_len + MIP_encap_hdr_len > MTU field in TCB) {
          frag_stat = WILLFRAG;
          if (the DF flag is set in the IP pkt)
             call icmp_error() to send ICMP_UNREACH_NEEDFRAG errmsg to CH;
       }
       else
          frag_stat = WONTFRAG;
    }
    else
       call icmp_error() to send ICMP_UNREACH_NET errmsg to CH;
 }
 if (TTL field in IP pkt < TTL field in TCB)
```

```
   call icmp_error() to send ICMP_TIMXCEED_INTRANS errmsg to CH;;
 return(frag_stat);
}
```

A.2 Foreign Agent Processing

This is shown as the dotted line in Figure A.1.

A.2.1 The MIP Input Module

This module is entered via the protocol switch table as shown in Figure 4.3(c). After determining the kind of encapsulation used, it restores/updates the state of the original IP header sent by the CH and calls `mip_output()` to do the output processing.

```
mip_input(m, ifp)
{
 find the outer IP header in the mbuf chain holding the IP packet
   ie.  call m_pullup() to yank atleast the equivalent of two IP headers
        into the first mbuf;
        call mtod() to get a pointer, ip, to the outer IP header;
 if (the DF flag in the outer IP header is set) {
    /* MIP encapsulation is being used */
    find the MIP encap header
      ie. after adjusting mbuf pointers to access data after outer IP header,
          call mtod() to get a pointer, mip, to the MIP encap header;
    test the S flag to determine whether the CH's address is there;
    verify the checksum of the MIP header;
    copy mip_src into outer IP header's ip_src field if S flag was set;
    copy mip_dst into outer IP header's ip_dst field;
    call mip_output(m, ifp, ip, mip, MIP_ENCAP);
 }
```

```
 else {
    /* IP encapsulation is being used */
    find the MIP encap header
      ie. after adjusting mbuf pointers to access data after outer IP header,
          call mtod() to get a pointer, in_ip, to the inner IP header;
    copy outer IP hdr's ip_ttl into inner IP's ip_ttl, and subtract one;
    call mip_output(m, ifp, ip, in_ip, IP_ENCAP);
 }
```

A.2.2 The MIP Output Module

This module strips the encapsulation header and transmits the packet locally. The IP address is a non-local one and would normally cause it to be routed back to the home network. However, the routing daemon would have already installed a local, direct route in the routing table by setting the RTF_GATEWAY to false. Additionally, it would have used an appropriate *ioctl* to set an entry in the ARP cache with the link address of the MN, determined from the MN's registration request at registration time. Hence we are also assured of proper IP to link-level address translation in the network-interface layer.

```
mip_output(m, ifp, outer, inner, encapsulation)
{
 switch(encapsulation) {
   case MIP_ENCAP: call m_copy() to copy the outer IP header into a new mbuf;
                   call m_adj() to strip the hdrs off the front of pkt;
                   prepend the saved IP hdr to bald pkt;
   case IP_ENCAP: call m_adj() to shave the outer encapsulating IP hdr;
 }
 call ip_output() to transmit the packet;
}
```

Appendix B

Kernel Probe Pseudocode

The logic and methodology for the kernel probes were derived from the work on TCP/IP performance by Papadapolous and Parulkar in [PP93].

B.1 Probe Activation

Probe activation occurs by detection of one of two protocols in the IP packet, that are not normally seen on the network to which the Agent is attached. These are a currently unused protocol (IP protocol number 100), PROBE, defined for the purpose of probe activation on Home Agents; and IP-in-IP, the encapsulation protocol used for tunneling, for the purpose of probe activation on Foreign Agents. The generic probe activation function may be represented by the fragment below. Such a fragment is introduced at the four queues specified in Figure 6.3.

```
...
if (IP packet protocol type is PROBE or IP-in-IP)
   call LOG_PACKET() to fill out a log record;
...
```

B.2 Packet Logging

```
LOG_PACKET() {
```

```
 raise interrupt level to block Ethernet interrupts
 if (we havent exhausted all the allocated logrecs)
    get ptr to curent head-of-log
    advance curent head-of-log
    store time in logrec field
    restore interrupt level to enable Ethernet interrupts
    fill in type and length fields
 else
    restore interrupt level to enable Ethernet interrupts
}
```

Appendix C

Glossary of Terms

Address Resolution Protocol, ARP: A protocol used for translating IP addresses to Ethernet addresses.

Agent Discovery: Home agents and foreign agents advertise their availability on each link for which they provide service. A newly arrived mobile node can send a solicitation on the link to learn if any prospective agents are present.

Care-of Address: The care-of address terminates the end of a tunnel toward a mobile node. Depending on the network configuration, the care-of address may be either dynamically assigned to the mobile node or associated with a foreign agent.

Decapsulation: At the care-of address, the enclosed datagram is extracted. When the mobile node receives packets sent to its own care-of address, it decapsulates its own datagrams. When the care-of address is associated with a foreign agent, the foreign agent decapsulates the datagrams. If the datagram is addressed to a mobile node which the foreign agent is currently serving, it will deliver the datagram to the mobile node.

Designated Multicast Service Provider, DMSP: The Home Agent selected by a Foreign Agent to forward multicast packets addressed to a specific group, under the mobile multicast scheme proposed in this thesis.

Directly connected: The state of having an IP address belonging to the same subnet as another IP node.

Encapsulation: Once a mobile node has registered a care-of address with its home agent, that home agent intercepts datagrams destined for the mobile node, builds another datagram with the intercepted datagram enclosed within, and forwards the resulting datagram to the entity at the care-of address. This process is also known as "tunneling".

Foreign Agent, FA: A router that assists a locally reachable mobile node that is away from its home network.

Gratuitous Proxy ARP: Extensions to ARP that allow packet interception to be performed by Home Agents in IETF Mobile IP.

Home Agent, HA: A router that keeps track of mobility bindings for mobile nodes that belong to the network shared by the home agent and the mobile nodes. It encapsulates datagrams for delivery to the mobile nodes while they are away from home.

Interface: A system's attachment point to a link. Interfaces are uniquely identified by IP unicast addresses.

Internet Control Message Protocol, ICMP: A protocol used to detect errors in IP.

Internet Engineering Task Force, IETF: A volunteer body of researchers and networking professionals which is responsible for the specification of protocols for use in the Internet.

Internet Protocol, IP: The Network layer protocol used in the TCP/IP protocol suite.

Link: A communication facility or medium over which systems can communicate at the link layer, i.e., the protocol layer immediately below IP. Examples of links are LANs (possibly bridged to other LANs), satellite channels, and point-to-point links.

Multicast: A mechanism for *one-to-many* communication in which the source transmits a single datagram, and the network performs the task of delivering that datagram to the multiple destinations.

Multicast interface: An interface to a multicast link, that is, an interface to a link over which IP multicast or IP broadcast service is supported.

Multicast link: A link over which IP multicast or IP broadcast service is supported. This includes broadcast media such as LANs and satellite channels, and also single point-to-point links.

Mobile Node, MN: A host that changes its point of attachment from one network or subnetwork to another.

Neighbouring: The state of having an IP address belonging to the same subnet as another IP node.

Registration: When the mobile node is away from home, it registers its care-of address with its home agent. Depending on its method of attachment, the mobile node will register either directly with its home agent, or through a foreign agent which forwards the registration to the home agent.

Router: A system that forwards IP datagrams. This does not include systems that, though capable of IP forwarding, have that capability turned off.

Subnet: Either a single subnet of a subnetted IP network [MP85] or a single non-subnetted IP network.

Transmission Control Protocol, TCP: The Transport layer protocol used in the TCP/IP protocol suite.

Tunneling: A mechanism to deliver packets across an internetwork.