Implementing Mobile IP Routing in 4.3 BSD Unix: A Case Study

Vineet Chikarmane*
Carey L. Williamson
Richard B. Bunt
Department of Computer Science
University of Saskatchewan
57 Campus Drive
Saskatoon, Saskatchewan, CANADA
S7N 5A9

June 27, 1996

Abstract

With increasing demands for mobility support in today's computing environments, routing packets to mobile hosts is a problem of considerable research and commercial interest. The introduction of mobility necessitates a change in the way routing is implemented in the Internet, and protocols to provide this routing effectively and efficiently are under active consideration. Mobile IP, a protocol intended for TCP/IP networks in general and the Internet in particular, has received a great deal of attention. The Internet Engineering Task Force has been working towards the development of a specification for Mobile IP. This emerging specification formed the basis of our investigation.

This paper describes the design, implementation, and performance evaluation of network layer modules to provide routing support for IETF Mobile IP in a 4.3 BSD Unix^{TM} system. In particular, we have fully implemented the Home Agent and Foreign Agent functionalities specified in IETF Mobile IP, and partially implemented the Mobile Node functionality required for mobility management. Decisions made during the design and implementation phases are discussed and defended. Our performance measurements provide evidence that the design and implementation are capable of comfortably handling the anticipated load imposed for the support of mobile hosts in the context of network technologies that are popular today, such as Ethernet.

*Presently at Glenayre Technologies in Vancouver, BC.

1 Introduction

Today's user community is demanding a level of mobility not previously anticipated by designers of current computer networks. Simultaneously, hardware technological advances have made inexpensive and powerful portable computers and wireless access technologies 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. A good survey of the challenges and opportunities in mobile computing is given in [Forman and Zahorjan 94]. A fully mobile environment should provide all the support necessary for applications to continue to operate in a transparent and seamless fashion. The research reported in this paper relates to one aspect of mobility support, namely support for location-independent internetwork access.

As mobile users roam about with their portable computing devices in hand, they would like to be able to connect to the internetwork from wherever they happen to be at the time they wish to access its services. However, present system and network software is unable to support this type of behaviour in a seamless way. For example, since IP routing [Postel 81a] is based on the network component of IP addresses, datagram routing fails when a mobile host moves to another network. Since routers cannot be expected to know the present location of every host, roaming must be managed by individual users, and system administrators, on an ad hoc case-by-case basis.

A number of proposals have come forward for network layer mobile host protocols to allow transparent interoperation across a TCP/IP internetwork. These proposals differ in the way in which host location information is represented and propagated to routers, and in the details of the mechanisms used to forward packets to mobile hosts.

An Internet Engineering Task Force (IETF) Working Group, called the Mobile IP group, was chartered in 1993 to coordinate efforts towards the specification of a protocol, **Mobile IP** [Perkins 95a], to handle host mobility in the Internet. Although this protocol is still in the process of being specified, it is being developed as an Internet standard. Through the spring of 1995, few (if any) implementations of IETF Mobile IP existed in the public domain. Implementation experience is an important step in the validation, and ultimately the acceptance, of this evolving standard.

Our goals for this project were a design specification and prototype implementation of IETF Mobile IP routing support in a 4.3 BSD Unix^{TM} environment, both to gain practical first-hand experience with Mobile IP and to provide a platform for future projects in mobility. This paper describes the various decisions that had to be made during the design and implementation phases, and presents the results of a performance evaluation of our prototype implementation. Since the routing of packets to mobile users would appear to present the greatest performance challenge, this was the focus of our performance study.

The remainder of this paper is organized as follows. Section 2 provides some background on IETF Mobile IP. Section 3 discusses the issues involved in implementing the IETF Mobile IP routing subsystem in 4.3 BSD Unix. Section 4 describes a set of experiments that we conducted to evaluate the performance of our implementation, and Section 5 provides some general conclusions.

2 Mobile IP as the Foundation for Mobility

2.1 Protocol Support for Mobility

As was mentioned in the introduction, several proposals have been presented for network layer protocols to support mobile hosts. Myles and Skellern [Myles and Skellern 93] evaluated three such proposals [Ioannidis 93, Perkins and Bhagwat 94, Teraoka *et al.* 92] based on considerations such as compatibility with existing infrastructure, route optimality, and migration procedures. Subsequent work [Myles and Perkins 93, Johnson 94, Perkins *et al.* 94] sought to address problems of the earlier proposals in areas such as robustness, scalability and security.

The IETF Mobile IP Working Group was chartered "to develop or adopt architectures and protocols to support mobility within the Internet"[1]. This group has been working on the specification of the Mobile IP protocol [Perkins 95a] to support host movement on the Internet. Where it is necessary to distinguish it from other Mobile IP proposals (such as Columbia University Mobile IP [Ioannidis *et al.* 91]), we will refer to the IETF protocol as IETF Mobile IP. The remainder of this section describes the (draft) protocol.

2.2 Overview of Mobile IP

As illustrated in Figure 1, there are four main functional entities in the Mobile IP specification. These are:

- **A Mobile Node (MN):** A Mobile Node is an entity, with a fixed IP address, that is capable of moving from one network to another in a TCP/IP internetwork. Each Mobile Node has a unique *home network*[2], that is designated by the *network identifier* in the Mobile Node's IP address. Any other network to which the Mobile Node may attach is called a *foreign network*.

- **A Correspondent Host (CH):** A Correspondent Host is an entity that is trying to send an IP packet to the Mobile Node, wherever the Mobile Node may currently reside.

[1] from the group's charter document, *IP Routing for Wireless/Mobile Hosts (mobileip)*, which is available on the IETF Mobile IP WWW site (URL ftp://software.watson.ibm.com/pub/mobile-ip/mobile-ip.html)

[2] We assume that the home network is a broadcast network, such as an Ethernet, where the Home Agent can always intercept packets addressed to the Mobile Node.

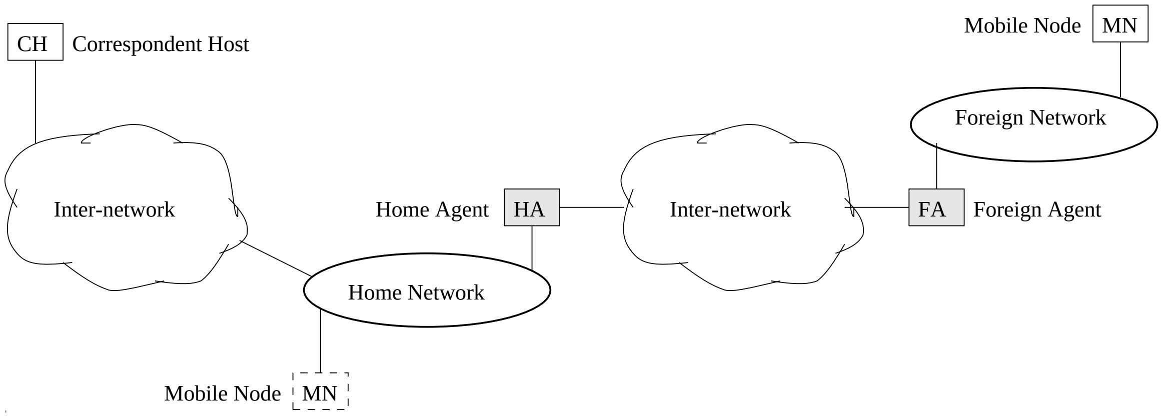

Figure 1: Mobile IP entities and their inter-relationships

- **The Home Agent (HA):** The Home Agent is an entity on the Mobile Node's home network that is responsible for forwarding IP packets to the Mobile Node when the Mobile Node is away from its home network.

- **The Foreign Agent (FA):** The Foreign Agent is an entity on the (foreign) network to which the Mobile Node is currently attached that has assumed the responsibility for forwarding IP packets to the Mobile Node as long as the Mobile Node is attached to this foreign network[3].

The Mobile IP protocol defines the interactions between these four entities. The Mobile Node is assigned an IP address on its home network, called the Mobile Node's *home address*. Packets from the Correspondent Host to the Mobile Node are always addressed to the home address of the Mobile Node, and are delivered to the home network through standard IP routing. When the Mobile Node is no longer at its home network, Mobile IP routing is used to route the packets to the Mobile Node's new location. This happens as follows. When the Mobile Node connects to a foreign network, it identifies and registers with a Foreign Agent, which also serves as its default router. During the registration process, the Foreign Agent provides the Mobile Node with a *care-of address*, defining the current location of the Mobile Node. The combination of the Mobile Node's home address and the care-of address is known as a *binding*. A timestamp originating from the Mobile Node (and updated with each reregistration) is associated with every home-address to care-of-address binding.

Each Foreign Agent maintains a list (*visitor list*) of all currently registered mobile nodes that are visiting the network served by the Foreign Agent. Each visitor list entry has an associated lifetime, which is negotiated during the registration process. A Mobile Node must reregister before this period has expired to ensure uninterrupted service from the Foreign Agent. The Foreign Agent establishes a permanent IP to link-level address mapping in its

[3]The Foreign Agent need not necessarily be the border router for the foreign network, as implied by Figure 1.

ARP (Address Resolution Protocol) cache by recording the MAC address of the Mobile Node from the registration request packet.

The Home Agent, attached to the Mobile Node's home network, maintains a *server cache* that identifies all mobile nodes that the Home Agent is configured to serve. The Home Agent maintains a current binding for all the mobile nodes in its server cache, reflecting their current locations, and hence when a Mobile Node registers with a Foreign Agent, the Mobile Node must also notify and register with its Home Agent.

Bindings and lifetimes are associated with all mobile nodes registered with a Home Agent. If the Mobile Node is at a foreign network, the Home Agent must arrange to intercept packets addressed to the Mobile Node and *tunnel* the packet, by *encapsulating* and sending it to the Foreign Agent that currently has the Mobile Node registered. When a Foreign Agent receives a tunneled packet addressed to a Mobile Node on its visitor list, the Foreign Agent *decapsulates* the packet and delivers it locally to the Mobile Node. In the case that a Mobile Node does not reregister upon expiry of the lifetime, the home agent assumes that the Mobile Node is connected to its home network.

3 Routing Subsystem Design

This section describes the routing subsystem that we designed and implemented in 4.3 BSD Unix to support Mobile IP. The subsystem requirements are given first, followed by an architectural overview and an examination of the design issues involved.

3.1 Design Requirements

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

- The Home Agent must agree to serve the Mobile Node, by inserting the Mobile Node's mobility binding into its server cache and intercepting packets destined for the Mobile Node.
- The Foreign Agent must agree to serve the Mobile Node, by inserting the Mobile Node's mobility binding into its visitor list.
- Rules must be established for the exchange and maintenance of location information.
- The actual routing of packets must be performed, using location information already in place.

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 Mobile Node connects to a foreign network. These procedures rely on the underlying location management procedures (the third requirement).

Finally, the fourth requirement deals with the actual routing of packets. Our focus in this paper is on the routing, which is likely to be the performance-critical part of Mobile IP. A more comprehensive discussion of the registration procedures and protocols used by the routing subsystem for location management is given in [Chikarmane 95].

3.2 Design Overview

The routing subsystem is intended primarily to support IETF Mobile IP, but it can support a forwarding mechanism for any mobile routing scheme as long as location information is made available. Our design and implementation are entirely in the context of the 4.3 BSD 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.

Three possible architectural organizations for the system were considered:

- a completely user-level implementation: Perform datagram processing in user space, with communication using IP raw sockets [Stevens 90]. Although it is straightforward to implement the mobility management functionality in user space, and this provides good user flexibility, there may be attendant performance problems. Furthermore, this would require that the entire IP protocol stack be implemented in user space as well.
- a completely in-kernel implementation: Perform both mobility management functions and packet routing using kernel routines. This improves performance, but may not be necessary. It also makes the implementation harder to debug.
- 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 split-level approach was selected[4] and the routing mechanism was implemented as shown in Figure 2. A user-level implementation of the Home and Foreign Agents was ruled out because of the context switch overhead for every packet processed, 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* [Leffler *et al.* 89] for header manipulation. A kernel implementation of mobility management was deemed unnecessary, since management packets and packets carrying location information are infrequent. Thus, location management was implemented in a user-level mobility daemon.

[4]The decision to split functionality between kernel and user-level is similar to that made for unicast and multicast routing in BSD systems.

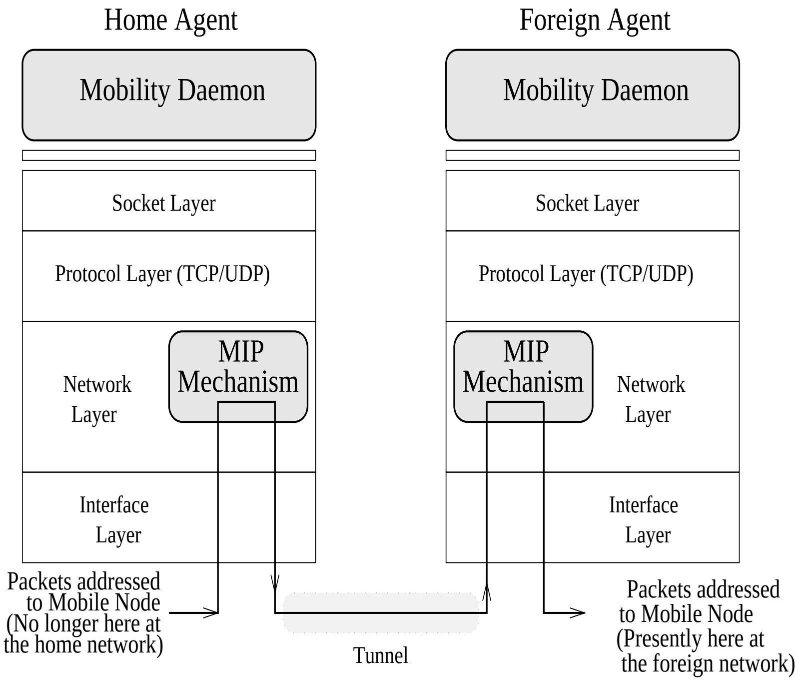

Figure 2: Architecture of the mobile routing software

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 mobile nodes must be served and how packets must be routed in the presence of mobility, while the routing subsystem actually effects the delivery of packets.

Given that the decision was 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. Accordingly, the system call interface of the kernel was extended by adding two new `ioctl` commands: `SIOCADDAT` to add entries into an in-kernel Agent Table, and `SIOCDELAT` to delete entries from the Agent Table. A single table is maintained for performing both Home and Foreign Agent functions.

The principal modules involved in mobile routing are shown in Figure 3, along with the specific paths taken by packets depending on whether the agent is tunneling the packet (i.e., functioning as a Home Agent), or processing already tunneled packets and forwarding locally (i.e., functioning as a Foreign Agent). The solid directed lines in the figure denote the path taken by packets being forwarded using conventional IP routing [Stevens 90].

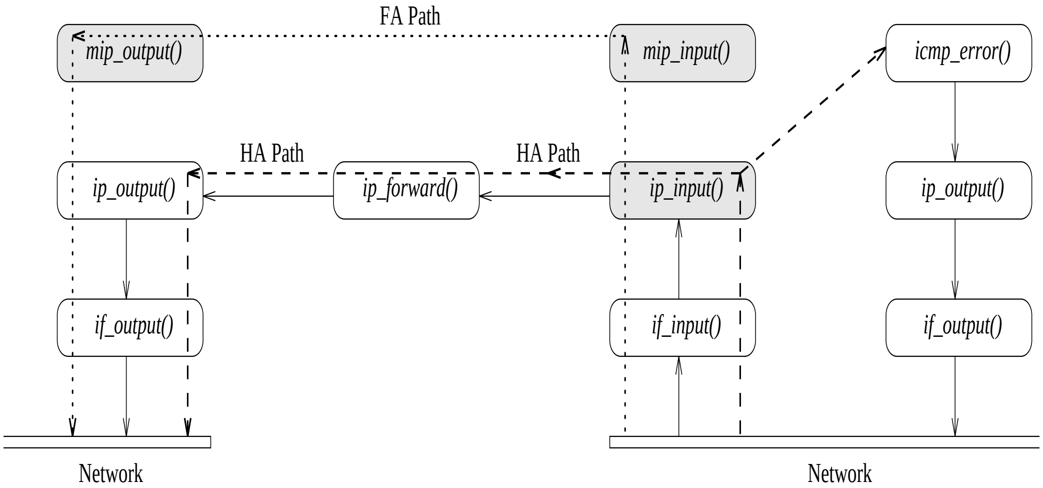

Figure 3: Paths taken by packets through the routing mechanism

3.3 Detailed Design Issues

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

3.3.1 Binding Cache Design

The kernel must maintain enough information to make routing and tunneling decisions. The mobility bindings of mobile nodes must be stored in a *server cache* for all mobile nodes for which the agent is serving as a Home Agent, and in a *visitor list* for all mobile nodes for which the agent is serving as a Foreign Agent. These caches may need to be searched/modified on the arrival of management packets. Furthermore, 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.

Fortunately, the routing tables maintained in Unix for supporting normal IP routing have all the needed facilities. Figure 4 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, as well as information on whether the destination is directly reachable (a direct route exists) or not (it is an indirect route), indicated by a special flag.

Our initial prototype implementation reused the existing routing tables by using an additional flag, `TUNNEL`, to indicate whether the route is to a Foreign Agent (i.e., the Mobile Node is away from home) at the far end of a tunnel. However, subsequent experience revealed that unexpected interactions with normal IP routing functions were possible[5]. To eliminate

[5]For example, we discovered that the Unix routing daemon `routed` could (dynamically) manipulate

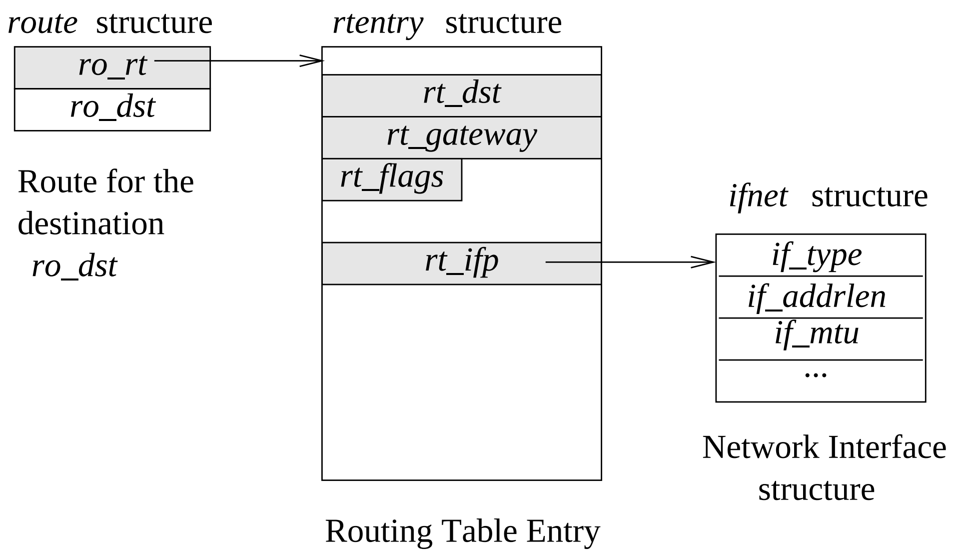

Figure 4: Data structures for routing

this possibility and in the interests of clean structure and flexibility, forwarding information for Mobile IP was implemented in a separate Agent Table (practically identical to the routing table) in the kernel.

3.3.2 Demultiplexing

From among the numerous packet types that must be processed by an Agent, three types of packets are of interest: *vanilla* packets, *management* packets and *tunneled* packets from correspondent hosts destined for mobile nodes.

Vanilla packets are addressed to the Mobile Node and are picked up by the Home Agent's network interface[6]. The Home Agent uses the destination field of the IP packet to differentiate between packets meant for itself, and those meant for mobile nodes that it is serving. Once identified, these packets are tunneled to the Foreign Agent.

Management packets may be sent and received using UDP sockets [Postel 80], and require no special support since network software to hand them up to the routing daemon already

the routing tables that we had (statically) set up for a Mobile IP experiment, completely disrupting the experiment. One solution, which we did not pursue, would be to modify `routed` to make it explicitly aware of our Mobile IP manipulations of the routing table. An easier solution was to use a completely separate routing table.

[6] Recall that the home network is assumed to be a broadcast network. Thus the Home Agent can Proxy ARP on behalf of the Mobile Node when the Mobile Node is away, with the result that all IP packets on the home network that are addressed to the Mobile Node get delivered to the link-level address of the Home Agent. Upon the return of the Mobile Node to the home network, the Home Agent (or the Mobile Node) should notify the other stations on the home network by sending a Gratuitous Proxy ARP, which then rebinds the Mobile Node's IP address to the Mobile Node's link-level address.

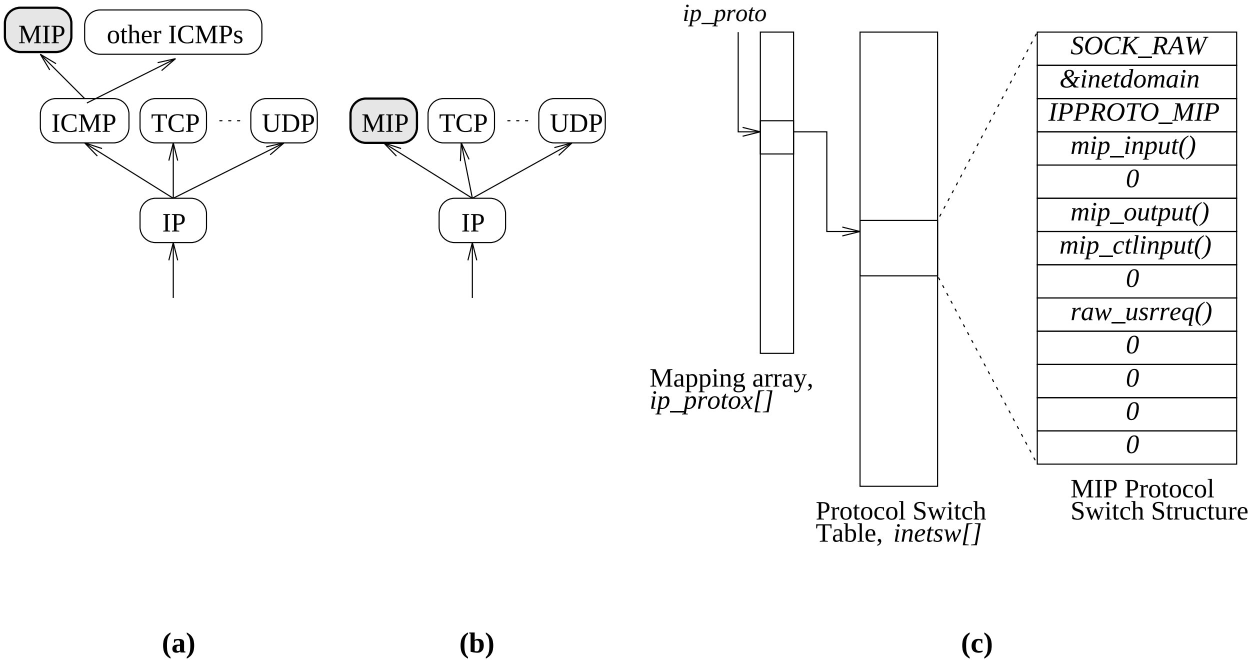

Figure 5: Demultiplexing and the Mobile IP protocol interface

exists.

Tunneled packets require special processing. In a very early draft proposal for Mobile IP, tunneled packets were compatible with the Internet Control Message Protocol (ICMP) [Postel 81b], 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 recognize 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 5(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. 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 5(b). As shown in Figure 5(c), the protocol field inside the IP header of the just-arrived packet is used to gain access to Mobile IP's input routine.

3.3.3 Fragmentation

The purpose of the encapsulation and tunneling of packets is to get "special" packets, such as Mobile IP packets or IP multicast packets, across an arbitrary internetwork that may or

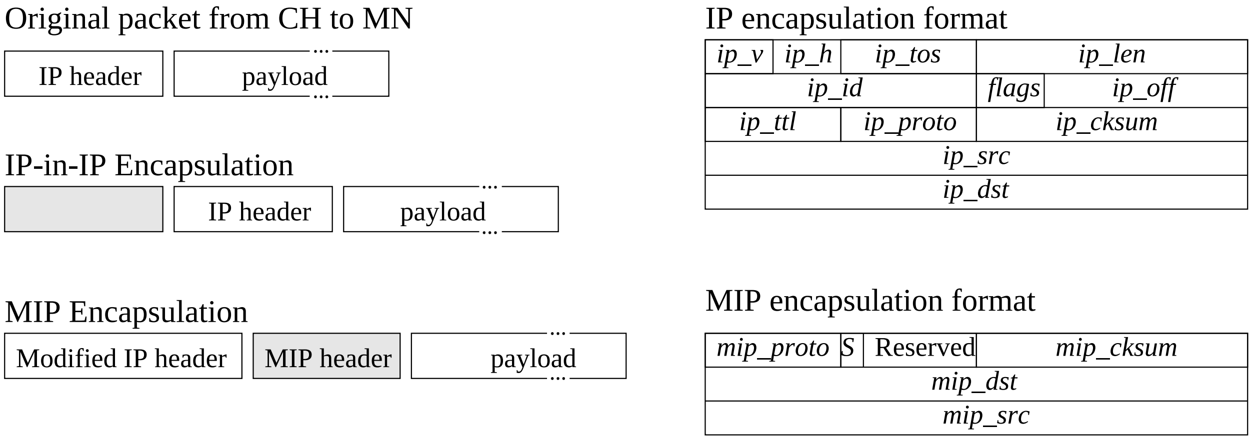

Figure 6: Encapsulation and the encapsulation headers used

may not support the protocols required by the "special" packet. In our context, for example, the Home Agent encapsulates (i.e., puts a new "wrapper" on the outside of) the original IP packet destined for the mobile node. This new wrapper is an IP packet header that lists the Home Agent as the source address and the Foreign Agent as the destination address. The data payload in the encapsulated packet is the original IP packet (header plus data). When the Foreign Agent receives the encapsulated packet, the Foreign Agent decapsulates (i.e., removes the wrapper from) the packet, and then determines that the payload is a self-contained IP packet to be delivered to the mobile node. The advantage of this approach is its transparency: no intermediate routers in the network between the Home Agent and the Foreign Agent are even aware that a "special" packet has traversed the network.

There are two alternative encapsulation formats currently supported by the Mobile IP protocol: IP-in-IP encapsulation and MIP encapsulation. These two formats are illustrated in Figure 6. The IETF has recently made IP-within-IP encapsulation mandatory to implement in Internet routers [Perkins 95b, Perkins 95c].

Unfortunately, encapsulation of a packet for tunneling increases its size (by at least the 20-byte minimum size IP header), creating the possibility of fragmentation if the enlarged packet exceeds the maximum transmission unit (MTU) of some link in the tunnel. Fragmentation is a serious problem in a TCP/IP internetwork because a fragmented packet is reassembled only at its final destination. This implies that the Mobile IP encapsulation header would be present only in the first IP fragment. The other packets (i.e., fragments) would not have the Mobile IP header, making proper delivery impossible.

Accordingly, we chose to force re-assembly of Mobile IP packets at Foreign Agents (before forwarding to the Mobile Node, to facilitate proper delivery) and at Home Agents (to necessitate only one encapsulation header per packet from a Correspondent Host to the Mobile Node). While requiring reassembly at Home Agents and Foreign Agents is somewhat controversial, it does simplify matters somewhat for the Mobile Node.

3.3.4 ICMP Back-Propagation

The back-propagation of ICMP error and control messages is an important consideration in the tunneling of packets. 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 tunneled packets have their IP header's `ip_src` (packet source address) field set to the outgoing interface of the Home Agent, ICMP software on all routers internal to the tunnel addresses messages to the Home Agent. Thus, all ICMP messages intended for the Correspondent Host (which sent the original packet to the Mobile Node) are sent to the Home Agent instead. Worse yet, the first 8 bytes of the IP packet causing the error contain only the first 8 bytes of the Mobile IP encapsulation header, and the `mip_src` field is not returned to the Home Agent. Thus, the Home Agent does not know to whom the ICMP message must be back-propagated, and any ICMP message back-propagated to the Correspondent Host will not contain the first 8 bytes of the IP payload of *its* packet that caused the error.

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, namely during the deployment and testing of a new routing protocol like Mobile IP.

The back-propagation problem has been addressed in the literature [Gilligan *et al.* 94, Simpson 94]. Simpson [Simpson 94], for example, has recommended (but not required) that the Home Agent maintain some form of *soft state*, keeping track of ICMPs received from the interior of a tunnel. This state 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, and the reachability of the end of the tunnel. Using this information, the Home Agent can return accurate ICMP messages in most cases.

Support for maintaining soft state has been incorporated into the design of our routing subsystem. It is enlightening to note, however, that soft state does not solve the problem, but serves only to ameliorate it since the Home Agent 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 mobile node, but will not be able to return round-trip times. Similarly, *traceroute*s to the mobile node would be blanked out inside the tunnel. ICMP back-propagation thus remains a problem for Mobile IP.

3.3.5 Maintaining Tunnel Soft State

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 7. The design must take into account the fact that several mobile nodes serviced by a Home Agent may be at the same care-of address (at the same foreign network). The soft state is updated with every ICMP message

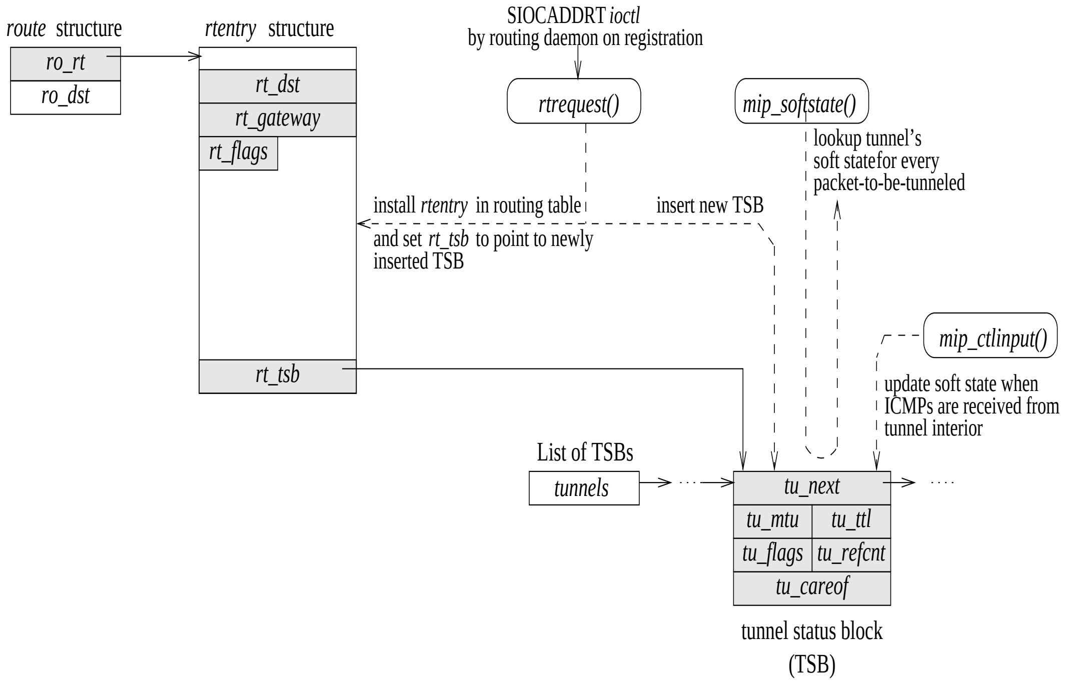

Figure 7: Storage and retrieval of soft state

returning from the tunnel, regardless of which mobile node the packet causing the error was addressed to. A new field, `rt_tsb`, was added to the routing (Agent) table entry that points directly to the TSB in question. This approach allows all information pertaining to mobile routing to be obtained by a single Agent table lookup in IP's input routine. The `mip_softstate()` routine allows this lookup to be made for each packet to be tunneled, and the `mip_ctlinput()` routine allows the soft state to be updated based on ICMP control messages that are received.

This organization of tunnel soft state enables fast lookups when tunneling packets to the mobile node, and fast updates when ICMPs arrive back to the Home Agent from the tunnel interior. Note that TSB list processing (insertion and deletion) occurs only during registration and registration expiry of mobile nodes.

3.4 Summary

This section provided a detailed look at the design of the routing functionality required to support 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 was also discussed, along with the tunnel state mechanism used to deal with it. The remainder of the paper considers the performance of our routing system design.

4 Performance of the Mobile IP Routing Subsystem

This section presents the results of our experimental evaluation of the performance of the Mobile IP routing subsystem prototype. As stated earlier, this subsystem is expected to be the most performance-critical part of Mobile IP, since the network packet traffic produced by location management (e.g., agent discovery, registration, re-registration) will be small compared to the packet traffic to and from mobile nodes. The following subsections describe the experiments that we designed, some issues related to taking the measurements, and the results of the experiments.

4.1 Performance Evaluation

Our performance study of the Mobile IP router implementation was conducted in the DISCUS Research Lab in the Department of Computer Science at the University of Saskatchewan. The Home and Foreign Agents were both HP 9000 series 400 workstations running HP-UX Version 8.0, a variant of 4.3 BSD Unix. The network was a 10 Mbps Ethernet. Our study focussed on delays in the IP and interface queues, the latency of the routing subsystem, and the total latency of the system. As was done in [Papadapoulos and Parulkar 93] our measurements were taken using *kernel probes*: small segments of code placed strategically in the operating system kernel to record information during the execution of the instrumented code. Kernel probes were inserted along the kernel path for Mobile IP packet processing. As shown in Figure 8, four queues labelled A, B, C and D lie along the path taken by packets that are routed to mobile nodes: 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 can be taken.

Two probes were associated with each queue: one that recorded information when data was appended to the queue, and the other that recorded information when data was removed from the queue. When activated, each probe recorded a timestamp, the amount of data passing through the reference point, and a type field to identify the location of the probe. Although the data recorded was minimal, it provided useful information about the behaviour of the 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.

In any such measurement, overhead is an issue since the introduction of measurement functionality in the kernel can introduce overheads that might have a significant impact on the accuracy of the measurements. This overhead must be kept to a minimum. Three sources of overhead are possible with these probes:

- overhead due to probe execution time: To minimize this, any processing other than the logging of a few important parameters was avoided.

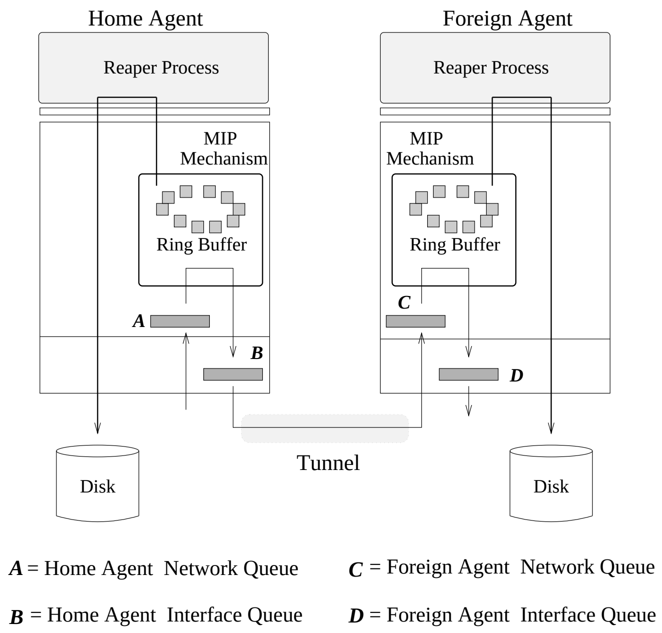

Figure 8: Routing subsystem measurement setup

- overhead due to storing of measurement data: 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 Mobile IP and the IP-in-IP encapsulation protocol. This simple test adds very little overhead to the packet processing.

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. In the HP-UX kernel used for our implementation, the `kernel_gettimeofday()` function provides time accurate to microsecond resolution by calling the assembly language routine `_get_precise_time`. This mechanism was found to provide the accuracy required for the timestamps that were recorded.

4.2 The Experimental Framework

The measurement experiments were conducted through the cooperative execution of several discrete components. These were:

- *Packet drivers* to pump packets into the Home and Foreign Agents to be processed.

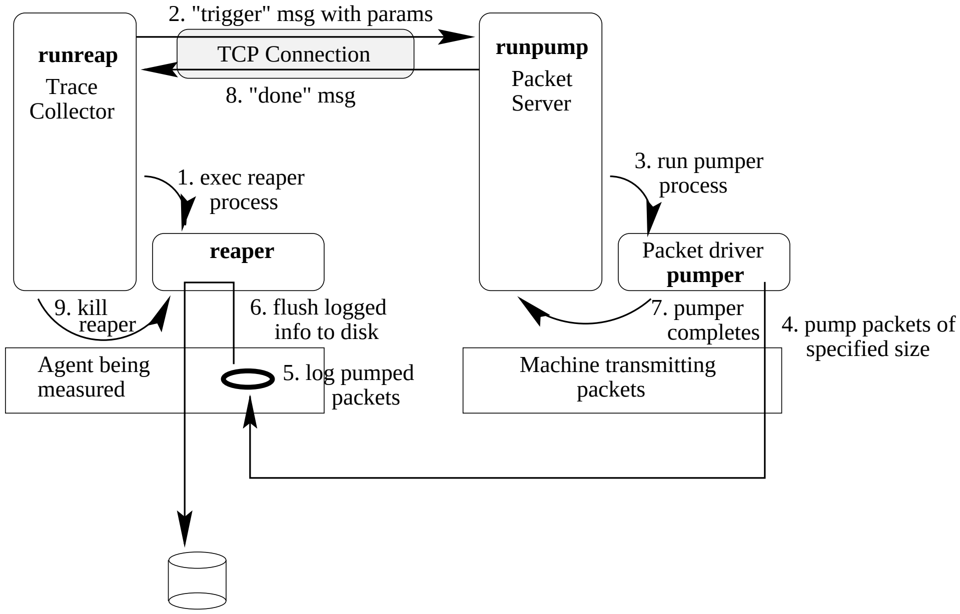

Figure 9: Measurement procedure and component interactions

- *Probes* to be activated when the relevant packets are being processed by the Home and Foregin Agents, in order to log the necessary information.

- A *reaper* process to flush the logged information into trace files.

A framework was needed to synchronize 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 provides a flexible platform for conducting the measurement experiments in a controlled and repeatable manner.

In these experiments the components that had to be synchronized resided on physically distinct machines. As shown in Figure 9, two processes communicating via a reliable TCP connection were used to synchronize 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 9.

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 packet driver transmits the indicated packets as fast as possible across the Ethernet LAN.

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. The measurement framework designed for these experiments is highly flexible and needs no intervention to perform the runs necessary to obtain the required measurements on the Home and Foreign Agents.

4.3 The Performance Experiments

Measurements were taken on both the Home Agent and the Foreign Agent. The sizes of the packets to be routed were chosen to embrace a number of issues of interest. The specific values chosen for the packet sizes are given in Table 1. In particular we included the minimum IP packet size (20 bytes), the maximum IP packet size as defined by the MTU of the link (1497 bytes for Ethernet), the largest size that did not cause the outgoing packet to be fragmented (1477 bytes), and the smallest size that caused the outgoing packet to be fragmented (1478 bytes). Other packet sizes were chosen to ensure an even distribution of packet sizes over the range from 20 to 1497. This range clearly spans right across the most common packet sizes observed for Internet traffic, namely 40, 72, and 552 bytes [Minshall 96].

Table 1: Packet sizes used in the experiments

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

4.4 Performance Results

The log records flushed into trace files on disk as described in Section 4.2 were processed and analyzed off-line to yield the measurements for the router implementation. These measurements are described in this section. All results reported here are the averages from 10 runs for each (Agent, packet size) combination. Each run used 1000 packets.

4.4.1 Delay

Measures of delay, obtained by using the timestamps recorded in each log record, are summarized in Figure 10. 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; and 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.

Queueing Delay at the Home Agent

The queueing delays at the Home Agent's IP and interface queues are shown in Figure 10(a). The queueing delay is almost constant with increasing packet size, indicating that there is no queue build up within the Home Agent. Each packet spends about 130 microseconds in 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.

Processing Delay at the Home Agent

The processing delays through the Home Agent are shown in Figure 10(b). Clearly, the routing delay (about 500 microseconds) accounts for most of the total delay (about 700 microseconds) with the queueing delays shown in Figure 10(a) accounting for the remainder (around 200 microseconds). The processing delay at the Home Agent is 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).

Queueing Delay at the Foreign Agent

The queueing delays at the Foreign Agent's IP and interface queues are shown in Figure 10(c). Again, the queueing delay is almost constant with increasing packet size. In fact, the queueing delays at the Foreign Agent match those at the Home Agent (130 microseconds in input IP queue, 50 microseconds for outgoing interface queue) very closely. There is no queue build up at the Foreign Agent. It should be noted that the 4.3 BSD stack does not copy data when it reassembles datagrams; it uses mainly `mbuf` manipulations instead. Hence the processing cost of packet reassembly is not expected to be significant.

Processing Delay at the Foreign Agent

The processing delays through the Foreign Agent are shown in Figure 10(d). The processing delay at the Foreign Agent is almost constant for all packet sizes, since no fragmentation occurs (fragmentation can only happen when a packet is being encapsulated, not decapsulated)[7]. The Foreign Agent processing delay is slightly lower than that measured at the Home Agent (450 versus 500 microseconds).

4.4.2 Packet Forwarding Rate

The packet forwarding rate was computed from the delay measures presented in the previous subsection. For example, the 700 microsecond processing time per packet at the Home Agent translates into a packet forwarding rate of approximately 1400 packets per second, regardless of the packet size (as long as no fragmentation occurs).

The resulting forwarding capacity (in bits per second) of our Mobile IP implementation thus scales linearly with packet size: from 0.7 Mbps for 64-byte packets, to 16.6 Mbps for 1477 byte packets. The fact that the latter value (16.6 Mbps) exceeds the 10 Mbps transmission capacity of an Ethernet LAN (i.e., 846 packets per second, for 1477 byte packets) means that our Mobile IP routing software will not be the bottleneck for forwarding large packets. Similar observations apply for the Foreign Agent.

The packet forwarding rate for the Home Agent drops abruptly when fragmentation occurs. Fragmentation occurs once the MTU is reached, resulting in the need to copy data and transmit two fragments on the outgoing interface for each incoming packet. In this situation, the maximum forwarding rate drops from 1400 packets per second (16.6 Mbps) for an incoming packet size of 1477 bytes (the maximum packet size without fragmentation on the outgoing interface) to 526 packets per second (6.2 Mbps) for an incoming packet size of

[7]Unfortunately, there are no performance results available for the packet reassembly process, since our kernel instrumentation was based on *incoming* IP packet size, not *outgoing* IP packet size.

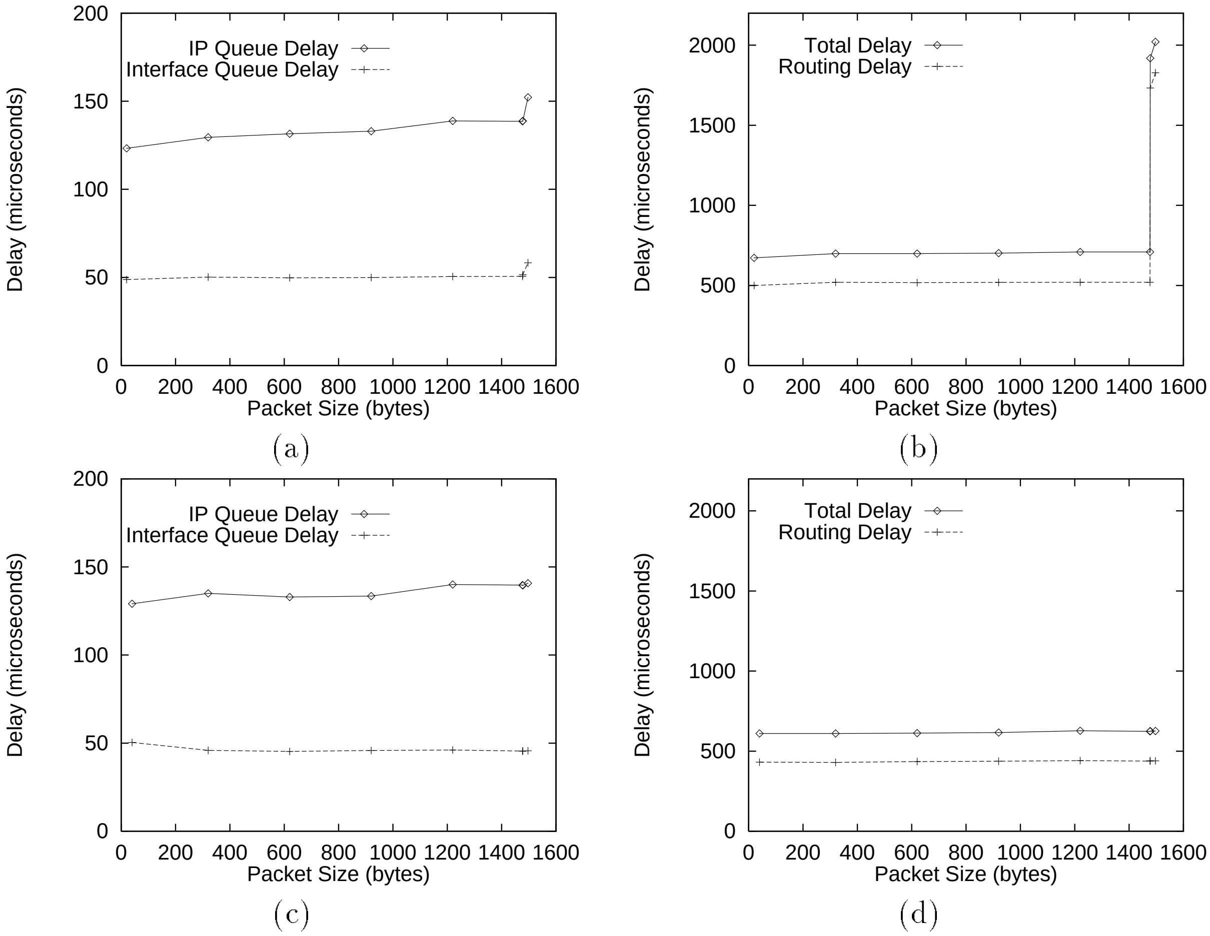

Figure 10: Delay measurements: (a) Queueing delay on the Home Agent; (b) Processing delay on the Home Agent; (c) Queueing delay on the Foreign Agent; (d) Processing delay on the Foreign Agent

1478 bytes (the minimum packet size that incurs fragmentation on the outgoing interface). Clearly, the performance impact of packet fragmentation is substantial.

4.5 Summary

The results presented from the delay measurements showed relatively constant delays in all four queues that were monitored, indicating no queue build up at either Agent, even when the packet pumper is transmitting at full Ethernet rate. The routing delay and total delay in both Agents were also relatively constant, except when packets were fragmented in the Home Agent. It appears that the overheads of the routing mechanism are within satisfactory limits.

The results presented on packet forwarding rate indicate the speed at which the Agents can successfully route packets to mobile nodes. For large packets, the forwarding rate of our routing mechanism exceeds the transmission capacity of the Ethernet (10 Mbps), and is much greater than the transmission capacity of wireless technologies in use today. This suggests that the Mobile IP packet processing in this implementation will not be the bottleneck for mobile network users, at least not in the near future.

5 Conclusions

This paper has described the design and implementation of routing support for IETF Mobile IP in 4.3 BSD Unix. Ours is among the first prototype implementations of IETF Mobile IP, and has served to identify several important issues in the design and deployment of Mobile IP. Although very preliminary, our performance analysis suggests that widespread deployment is feasible but indicates areas of concern that warrant further study.

Several general conclusions can be drawn from this research. First, the design choices made in implementing mobile routing support were effective. Kernel-level implementation of the routing mechanism is essential for performance reasons, while user-level support for mobility management adds greater flexibility to track the (changing) Mobile IP protocol specification. Second, the fragmentation overhead due to the tunneling which is used in Mobile IP is significant. Efforts should be undertaken to reduce this overhead, or to avoid fragmentation whenever possible. Unfortunately, fragmentation problems are likely to be at their worst on low-bandwidth wireless networks, where MTU's are typically small[8]. Finally, the performance measurements of our Mobile IP routing subsystem indicate that it can comfortably handle the anticipated load imposed for the support of mobile hosts. The forwarding rate of the Agents compares favourably with current network technologies, such

[8] Higher layer protocols, such as TCP, could likely be "tuned" for the wireless environment to prevent this problem (e.g., by setting the Maximum Segment Size (MSS) properly, or by using a proxy between the Mobile Node and the Foreign Agent).

as Ethernet, and so the Agents, implemented as described in this paper, will not constitute a bottleneck in the network infrastructure for the support of mobile hosts.

In conclusion, we found that Mobile IP provides the intended support for mobile users in an effective and efficient manner, and integrates relatively easily into the existing TCP/IP base in BSD Unix. We look forward to its widespread availability in the near future.

Acknowledgements

Financial support for this research came from several sources, including the Natural Sciences and Engineering Research Council of Canada (NSERC) through research grants OGP0003707 and OGP0121969, Telecommunications Research Laboratories (TR*Labs*), and IBM Canada's Centre for Advanced Studies. This support is gratefully acknowledged, as is the support (and patience) of our colleagues in the *DISCUS* lab at the University of Saskatchewan throughout our implementation and evaluation efforts. Finally, we acknowledge the helpful comments of the referees who reviewed this paper.

References

[Chikarmane 95] V. Chikarmane, *Network Support for Mobile Hosts in a TCP/IP Internetwork*, M.Sc. Thesis, Department of Computer Science, University of Saskatchewan, Saskatoon, Canada, August 1995.

[Forman and Zahorjan 94] G. Forman and J. Zahorjan, "The Challenges of Mobile Computing", *IEEE Computer*, Vol. 27, No. 4, pp. 38-47, April 1994.

[Gilligan *et al.* 94] R. Gilligan, E. Nordmark, and B. Hinden, "IPAE: The SIPP Interoperability and Transition Mechanism", Internet Draft, March 1994.

[Imielinski and Badrinath 94] T. Imielinski, and B.R. Badrinath, "Mobile Wireless Computing", *Communications of the ACM*, Vol. 37, No. 10, pp. 18-28, October 1994.

[Ioannidis 93] J. Ioannidis, *Protocols for Mobile Internetworking*, Ph.D. Thesis, Columbia University, New York, New York, 1993.

[Ioannidis *et al.* 91] J. Ioannidis, D. Duchamp and G. Maguire, "IP-based Protocols for Mobile Internetworking", *Proc. ACM SIGCOMM '91 Conference: Communications Architectures and Protocols*, Zürich, Switzerland, pp. 235-245, September 1991.

[Johnson 94] D. Johnson, "Scalable and Robust Internetwork Routing for Mobile Hosts", *Proc. 14th International Conference on Distributed Computing Systems*, June 1994.

[Leffler *et al.* 89] S. Leffler, M. McKusick, M. Karels, and J. Quarterman, *The Design and Implementation of the 4.3 BSD UNIX Operating System*, Addison-Wesley Publishing Co. Inc., 1989.

[Minshall 96] G. Minshall, information available from Web site at `http://www.nlanr.net/NA/Learn/packetsizes.html`

[Myles and Perkins 93] A. Myles and C. Perkins, *Mobile IP Extensions (MIP)*, Internet Draft, October 1993.

[Myles and Skellern 93] A. Myles and D. Skellern, "Comparison of Mobile Host Protocols for IP", *Internetworking Research and Experience*, Vol. 4, No. 4, pp. 175-194, December 1993.

[Papadapoulos and Parulkar 93] C. Papadapoulos and G.M. Parulkar, "Experimental Evaluation of SUNOS IPC and TCP/IP Protocol Implementation", *IEEE/ACM Transactions on Networking*, Vol. 1, No. 2, pp. 199-216, April 1993.

[Perkins 95a] C. Perkins, *IP Mobility Support*, draft-ietf-mobileip-protocol-11.txt, Mobile IP Working Group, Internet Draft, July 1995.

[Perkins 95b] C. Perkins, *Minimal Encapsulation Within IP*, Internet Draft, July 1995.

[Perkins 95c] C. Perkins, *IP Encapsulation Within IP*, draft-ietf-ip4inip4-01.txt, Internet Draft, October 1995.

[Perkins and Bhagwat 94] C. Perkins and P. Bhagwat, *A Mobile Networking System Based on Internet Protocol*, *IEEE Personal Communications*, Vol. 1, No. 1, pp. 32-41, March 1994.

[Perkins *et al.* 94] C. Perkins, A. Myles, and D. Johnson, "IMHP: A Mobile Host Protocol for the Internet", *Computer Networks and ISDN Systems*, Vol. 27, No. 3, pp. 479-491, December 1994.

[Postel 80] J. Postel, "RFC-768: User Datagram Protocol", Request for Comments, Internet Network Information Center, August 1980.

[Postel 81a] J. Postel, "RFC-791: Internet Protocol", Request for Comments, Internet Network Information Center, September 1981.

[Postel 81b] J. Postel, "RFC-792: Internet Control Message Protocol", Request for Comments, Internet Network Information Center, September 1981.

[Simpson 94] W.A. Simpson, *IP Mobility Support*, Internet Draft, Mobile IP Working Group, IETF, March 1994.

[Stevens 90] W.R. Stevens, *UNIX Network Programming*, Prentice-Hall Inc., 1990.

[Teraoka *et al.* 92] F. Teraoka, K. Claffy, and M. Tokoro, "Design, Implementation, and Evaluation of Virtual Internet Protocol", *Proc. 12th International Conference on Distributed Computing Systems*, Pittsburgh, PA, pp. 170-177, June 1992.