Report Number: WUCS-98-11

1998-01-01

Fault-Tolerant Mobile IP

Rajib Ghosh and George Varghese

We describe mechanisms to enhance the reliability and performance of Mobile IP. In Mobile IP today home agents and foreign agents are single points of failure and potential performance bottlenecks. For example, a home agent crash can lead to communication failure if the mobile is away from home. In this paper we describe new mechanisms to allow redundant home and foreign agents. Redundant agents can take over from each other in case of failure, and also split load amongst themselves. Our mechanisms are simple, transparent to existing mobile nodes, and compatible with the existing Mobile IP specification. We have... **Read complete abstract on page 2.**

Fault-Tolerant Mobile IP

Rajib Ghosh and George Varghese

Complete Abstract:

We describe mechanisms to enhance the reliability and performance of Mobile IP. In Mobile IP today home agents and foreign agents are single points of failure and potential performance bottlenecks. For example, a home agent crash can lead to communication failure if the mobile is away from home. In this paper we describe new mechanisms to allow redundant home and foreign agents. Redundant agents can take over from each other in case of failure, and also split load amongst themselves. Our mechanisms are simple, transparent to existing mobile nodes, and compatible with the existing Mobile IP specification. We have implemented our scheme on top of an existing Mobile IP implementation, and our implementation adds about 700 lines of code. Our paper describes and defends our design choices, the synchronization protocols used, and the implementation of our fault-tolerant protocol.

Fault-Tolerant Mobile IP

Rajib Ghosh
George Varghese

WUCS-98-11

April 29, 1998

Department of Computer Science
Campus Box 1045
Washington University
One Brookings Drive
St. Louis, MO 63130-4899

Abstract

We describe mechanisms to enhance the reliability and performance of Mobile IP. In Mobile IP today home agents and foreign agents are single points of failure and potential performance bottlenecks. For example, a home agent crash can can lead to communication failure if the mobile is away from home. In this paper we describe new mechanisms to allow redundant home and foreign agents. Redundant agents can take over from each other in case of failure, and also split load amongst themselves. Our mechanisms are simple, transparent to existing mobile nodes, amd compatible with the existing Mobile IP specification. We have implemented our scheme on top of an existing Mobile IP implementation, and our implementation adds about 700 lines of code. Our paper describes and defends our design choices, the synchronization protocols used, and the implementation of our fault-tolerant protocol.

Keywords: Internet Protocol, Mobile IP, Mobility Agents, Mobile Host, Home Agent, Foreign Agent, ARP, Gratuitous ARP, IP Encapsulation & Tunneling, ICMP.

This work was supported by National Science Foundation under grant NCR-9628145 and the ONR Young Investigator Award.

Fault-Tolerant Mobile IP

Rajib Ghosh
rajib@ccrc.wustl.edu
+1 (314) 935-4163

George Varghese
varghese@askew.wustl.edu
+1 (314) 935-4963

Abstract

We describe mechanisms to enhance the reliability and performance of Mobile IP. In Mobile IP today home agents and foreign agents are single points of failure and potential performance bottlenecks. For example, a home agent crash can can lead to communication failure if the mobile is away from home. In this paper we describe new mechanisms to allow redundant home and foreign agents. Redundant agents can take over from each other in case of failure, and also split load amongst themselves. Our mechanisms are simple, transparent to existing mobile nodes, amd compatible with the existing Mobile IP specification. We have implemented our scheme on top of an existing Mobile IP implementation, and our implementation adds about 700 lines of code. Our paper describes and defends our design choices, the synchronization protocols used, and the implementation of our fault-tolerant protocol.

Keywords: Internet Protocol, Mobile IP, Mobility Agents, Mobile Host, Home Agent, Foreign Agent, ARP, Gratuitous ARP, IP Encapsulation & Tunneling, ICMP.

1. Introduction

Mobile computing is becoming ubiquitous. The past few years have seen a proliferation of portable computers and laptops. A *Mobile Host* (e.g., a laptop) may be in use continuously through a wireless network interface as it is being carried from one place to another, or it may simply be disconnected from one interface and plugged into another interface at a new location.

However, current Internet routing protocols (IP, OSI, etc.) require the network address to change when a host moves to a new location. This is inconvenient for laptops and other mobile stations. The Mobile IP Protocol (MIP) [Per96b] handles this problem using *Mobility Agents*. Mobility agents keep track of *mobile hosts* and are responsible for routing packets to them. Agents, however, use static IP [Pos81] for routing their packets. Mobile IP is simple in the sense that it uses the existing mechanisms of IP Encapsulation [Sim95, Per96a, Per96c], ICMP messages [Dee91] and ARP [Plu82]. Mobile IP uses static IP routing, but leaves the latter completely unaffected.

Overview of Mobile IP: In Mobile IP [Per96b], a source S sending a packet to a mobile M sends the packet to a fixed IP address for M. Because of the way IP routing works, this packet is routed to the "home" subnet on which M resides when it is not mobile. If M is away from home on a foreign

subnetwork (Figure 1) a *Home Agent* for the mobile host intercepts packets addressed to M. The home agent then *encapsulates* the packet and *tunnels* the packet to a designated *Foreign Agent* on the foreign subnetwork on which M currently resides. The Foreign Agent *decapsulates* the packet and transmits it on the foreign network (see Figure 1). If the mobile is using a so-called *co-located care-of address*, the home agent can send the encapsulated packet to the mobile directly, bypassing the foreign agent. If M is using a co-located address, M will decapsulate the packet instead of the foreign agent.

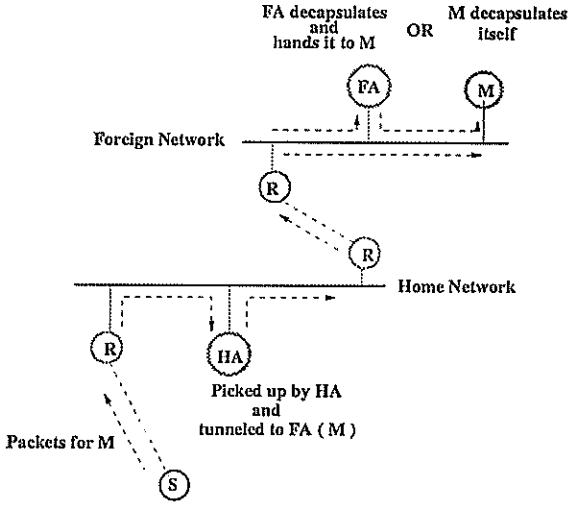


Figure 1: Mobile IP. Source S sends a packet to mobile M. The home agent HA intercepts the packet, encapsulates and tunnels it to the foreign agent FA. FA then decapsulates it and hands it to M. Alternatively, if M is using a co-located care-of address, HA sends the encapsulated packet to M directly, and M decapsulates it itself.

For the rest of this paper, we will refer to Mobile IP using the acronym MIP. In more detail, the MIP protocol works as follows.

- Mobility Agents (i.e. home (HA) and foreign (FA) agents) frequently advertise their presence via Agent Advertisement messages. A mobile M receives these advertisements and determines whether it is on its home network or a foreign network.
- When M detects it is at home, it operates without mobility services. When returning to its home network, it deregisters with HA through the exchange of Registration Request and Reply messages.
- When M detects that it has moved to a foreign network, it obtains a *care-of* address on the foreign network. The care-of address can be either be obtained from agent advertisements (a foreign agent care-of address), or by some other mechanism like DHCP [Dro93] (a co-located care-of address). M then registers its new care-of address with its home agent HA through Registration Request and Reply messages, possibly via its foreign agent FA.
- Datagrams sent to M are intercepted by HA, encapsulated and tunneled to the mobile's care-of address. They are received at the tunnel endpoint by FA (or by M itself), decapsulated and handed to M. Datagrams sent by M are routed to their destination by static IP routing.

A care-of address is an address to which datagrams can be routed using standard IP routing. A foreign agent care-of address is one provided by the foreign agent through its agent advertisement messages. In this case, the foreign agent decapsulates the packets and hands them to the mobile.

A co-located address is a care-of address acquired by the mobile as a local IP address on the foreign network. It may be obtained temporarily (e.g., through DHCP [Dro93]), or it may be a long term address reserved for its use while visiting the foreign network. In such a case, the mobile decapsulates its own packets. Thus, a foreign agent is not essential to the working of the protocol. Figure 1 illustrates the routing of a datagram when the mobile M is away from home.

As with any new mechanism, MIP is still in an experimental phase. However, once it becomes commercial, service providers will have to ensure reliability and performance. Home agents in MIP are single points of failure. If the home agent crashes, all communication to the mobile is disrupted. The mobile will eventually discover this fact because it fails to get acknowledgements for its registration messages. However, the time period between registration requests is deliberately set to a fairly large interval to avoid frequent traffic across the Internet. Thus the most obvious way of detecting home agent failures may be too slow.

A foreign agent failure is less critical in the sense that a mobile can detect failure when it fails to hear an Agent Advertisement from its foreign agent. When this happens, the mobile can register with another foreign agent if a backup agent is available.

Further, with the growing importance of mobility, agents someday might have to serve numerous mobiles. Having a single home or foreign agent serve all mobile nodes on a network can affect the reliability and performance of MIP. Besides the reliability problems we have alluded to, too much load on a single agent could be detrimental to MIP performance. Thus, mobility agents could be possible performance bottlenecks for mobile IP in the near future.

Related Work: The MIP literature [Joh91, TCT92, MS93, PMJ94, Joh94], so far, pays very little mention to *fault-tolerance*. [MS93] compares four IP based Mobile Host Protocols; according to this paper, all existing schemes require mobility agent duplication to handle component loss. The *Virtual Internet Protocol* [TCT92] paper does mention handling failures using duplication of the *Address Mapping Table (AMT)* entries at multiple gateways. However, it still requires duplication of the home gateway; the paper[TCT92] does not describe at all how this duplication can be achieved.

After beginning this project, we learned of work being done by the Secure Mobile Networking project at Portland State University. The primary goal of this project has been to integrate security and network layer mobility into real systems. However, their Sixth Quarterly Report[BM97] describes some preliminary ideas to handle redundancy of home agents. In HARP, *Home Agent Redundancy Protocol* [BM97], both the HARP peers maintain routing tunnels to the mobile host and to each other. The *harpies* (HARP peers) are usually on different networks, but they present a consistent mobile IP subnet to the world; i.e., they advertise a single subnet to the routing domain. When the mobile host is at home, one harpy tunnels to the other where it is actually present. Harpies use "Hello" messages to monitor each other. A harpy forwards MIP registrations to its peer, and upon boot obtains the entire routing table from its statically configured peer.

When comparing the work in [BM97] to our approach, we find both similarities and differences. Our approach to synchronizing bindings (by having registration messages forwarded to backup home agents, and by obtaining bindings on startup if necessary) is similar to the approach in [BM97]. While we had invented our approach based on registration forwarding before we saw the work in [BM97], we did decide to use the [BM97] approach of obtaining bindings on startup using TCP. However, we note that this is a common approach used in the Internet world for synchronizing state information for protocols such as DNS and BGP.

On the other hand, the approach in [BM97] allows the home agent peers to be on different networks for redundancy. This requires some complexity in terms of a virtual subnetwork that allows the mobile agent to be transparent to the presence of the multiple home agents. It also implies that packets sent to the mobile can be routed (and intercepted) by any of the home agents,

depending on the dynamic nature of IP routing. We avoid this potential complexity (while losing some of the potential benefits) by only allowing our multiple agents to be on the same network.

Secondly, the work in [BM97] only allows one backup agent, while our protocol was explicitly designed to allow multiple backup agents (and yet allow a simple implementation in the common case of a single backup agent). Thirdly, the work in [BM97] does not mention the use of load balancing. We also note that the initial report in [BM97] only describes the basic idea in 3 pages. We have found a number of subtle details after actually specifying and implementing the complete protocol.

Outline of Paper: This paper describes in detail how our fault tolerant MIP protocol works. In a nutshell, we provide fault tolerance by having multiple mobility agents. The peer agents constantly monitor each other; when one of them crashes, another agent takes over its job. It appears to be a waste of resources for a machine to be only a backup agent. Thus we also allow *load-splitting*. The primary home agent (which is the only home agent a given mobile is aware of) can different for each mobile, which allows load balancing. We have also kept the changes to the protocol almost completely transparent to existing mobile hosts (Section 3.3 describes a small detail regarding detecting movement.) Thus, when an agent crashes, and its peer takes over, the mobile host does not notice any change (except a small discontinuity in its packets). Our FT protocol is completely compliant with the current MIP specifications [Per96b]. Section 2.1 describes our design goals in detail.

Section 2 discusses our design approach; it describes our goals, how we deal with agent failure and recovery, and how we synchronize bindings between agent peers. Section 3 describes our protocol in detail. We also implemented our protocol on top of an existing mobile IP implementation which we obtained from SUNY Binghampton. Section 4 describes our implementation. Our protocol can be extended to allow multicasting and to incorporate security and authentication features. We discuss these extensions in Section 5. We state our conclusions in Section 6.

2. Design Approach

This section discusses our approach to designing our fault-tolerant (FT) mobile IP (MIP) protocol. We first describe our goals; we then describe how we handle agent failure and recovery in the light of these goals. We discuss how we synchronize mobility bindings between agent peers. We also discuss alternative approaches and explain why we discarded them.

2.1. Goals

When designing our protocol, we had the following simple goals in mind. Our goals defend our design against other possible approaches.

Fault Tolerance: Our primary goal is to allow multiple home agents and foreign agents such that if one agent fails another can perform its functions. As discussed earlier, mobility agents are single points of failure in the system and their failure leads to disruption of communication with the mobile.

Load Splitting: With the increase in the number of mobiles, having a single agent serve them all could slow down mobile communication. An obvious outcome of multiple agents is to have them share load amongst themselves. This would allow future expansion of mobile IP and would enhance its performance.

Transparency: Another important criterion in our design was to keep the current mobile host protocol unchanged. This allows our protocol to be implemented by replacing software only at the

agents. All existing mobile hosts remain unchanged. Thus, when an agent crashes and its peer takes over for it, the mobile notices no change except for a small discontinuity in its packets.

Simplicity and Compatibility: Our protocol adheres to the simple design philosophy of mobile IP. We continue to use existing mechanisms in mobile IP like IP Encapsulation [Sim95, Per96a, Per96c], ICMP [Dee91], and ARP [Plu82]. We use the existing Registration Request and Reply messages to distribute mobility bindings among agent peers. Finally, an agent uses TCP [Ste94, WS94] to obtain its bindings when it comes up after a crash (see Section 2 for details). Gratuitous ARP is used by an agent to takeover from its failed peer. Our design is also compliant with the current specifications of mobile IP [Per96b].

Efficiency: We tried to keep the extra messages in our protocol to a minimum. Our design does not increase Internet traffic (e.g., it leaves unchanged the registration messages sent between the mobile and its current home agent). Mobility bindings are distributed and obtained locally over the LAN. A mobile registers with only a single agent as before. Our protocol has no explicit "Hello" messages between agent peers (we have agents listen to each others advertisements instead); despite this, our protocol allows recovery at small time periods (a few seconds).

2.2. Redundant Home Agents

One possible way of handling multiple home agents would be to have the mobile register with all of them simultaneously, and have all of them maintain routing tunnels to its care-of address. Alternatively, mobiles themselves, can find out if their primary home agent has failed (from registration timeout), and then attempt to register with a secondary home agent. The above approaches require cooperation from the mobile itself, and is not transparent to existing mobile nodes.

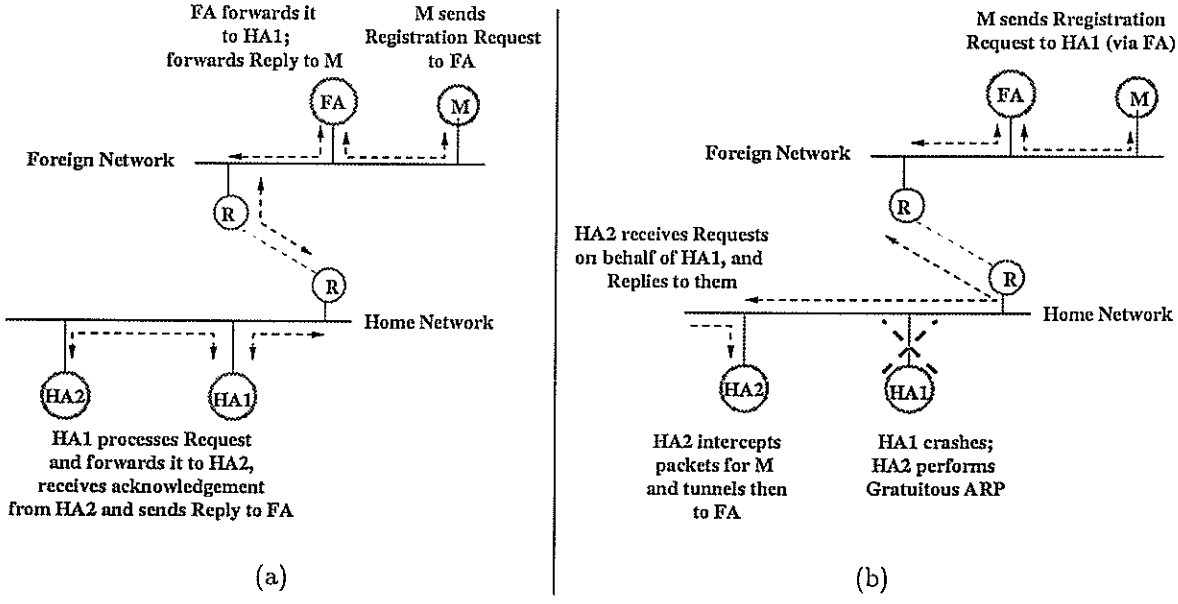


Figure 2: (a) Registration Forwarding among Home Agent peers; $HA1$ forwards Registration Request to $HA2$, a Reply is sent on receiving an acknowledgement from it. This ensures that the mobility binding is reliably recorded at the peer. (b) Home Agent Crashes; $HA1$ fails, $HA2$ performs Gratuitous ARP and replies to all registration requests to $HA1$. It also intercepts M's packets and tunnels then to FA.

Further, registration timeouts are in the order of minutes (usually 2 - 3 minutes), and we would like much faster recovery. Note that registration lifetime is larger so as to decrease the number of registration messages which flow between the mobile and its home agent across the Internet. Agent Advertisements, on the other hand, are sent out locally; hence their lifetime is usually in seconds. Thus, we would like our protocol to respond to advertisements (or rather their absence) to detect failure in seconds. Most applications can handle a few seconds of discontinuity. Having the mobile register with all the home agents would also increase the Registration Request and Reply messages flowing across the Internet. We use local registration forwarding instead to avoid this.

Figure 2(a) depicts a typical scenario in our design. The mobile M registers with only one home agent ($HA1$) as before. $HA1$ processes it and then forwards it to $HA2$, who stores the mobility binding and sends an acknowledgement back. $HA1$ sends its reply to M only on receiving this acknowledgement. This ensures that M's registration is recorded reliabily at $HA2$. In our design, the multiple home agents are on the same network. It allows the home agents to monitor each other by snooping on their Agent Advertisement messages.

Agents advertise their services every few seconds (2-3 seconds in most implementations). This obviates the need for special "Hello" messages, and also allows faster recovery. When $HA1$ crashes (see Figure 2(b)), $HA2$ performs gratuitous ARP to takeover the functions of $HA1$. Thus, our protocol is transparent to existing mobiles. After the crash, $HA2$ advertises for $HA1$ so that the mobile does not detect a move.

When $HA1$ comes up after a crash (Figure 3), it first asserts itself through gratuitous ARP, and then uses TCP to obtain its mobility bindings from $HA2$. Using TCP has the advantage that TCP takes care of reliability and resequencing (the list of bindings to be obtained might be huge). Note however that TCP works over static IP (between peer agents on the same network), and so we do not have a bootstrapping problem. Using TCP is preferable to inventing (yet another) protocol to reliably obtain bindings.

If there are more than two home agents, we need some kind of arbitration mechanism to decide which agent takes over if one fails. We have every home agent monitor every other agent, and use the monitoring information to maintain an *active peer set*. An active peer set is the set of agents which are currently *alive*. When an agent deems one of its peers to have failed, it looks into its active peer set (which now excludes the failed agent, but includes itself), and applies a boolean function on the peer set. If the function returns *true*, the agent decides to take over. For example, a simple function might be "Do I have the smallest address?".[1] The boolean function should return *true* for one and only one agent. The home agent who decides to take over, now advertises for its failed peer in its own Agent Advertisements.

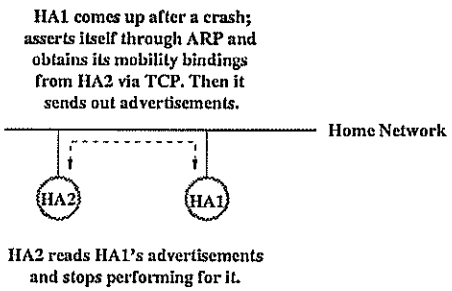


Figure 3: Home Agent comes up after a crash; $HA1$ asserts itself through Gratuitous ARP and obtains its mobility bindings from $HA2$ via TCP. The mobile might notice a brief loss of packets during this interval.

[1] More complex functions can allow more load splitting when failures occur.

Home agents advertise for failed peers they are substituting for. Thus, a mobile detects it is at home if it receives an Agent Advertisement containing its home agent's address. This is a minor deviation from RFC 2002 which states that a mobile should receive an advertisement only from its home agent. If mobiles are configured with home agents, we can guarantee static load-splitting by configuring the mobiles with different home agent addresses. If mobiles use dynamic home agent address resolution as described in [Per96b], they can randomly choose one from among the different replies they receive from the multiple home agents.

2.3. Redundant Foreign Agents

In mobile IP today, foreign agents are not as critical as home agents. If a mobile uses co-located care-of address, foreign agents are not required at all. If mobiles use foreign agent care-of address, we need redundancy to take care of failures. Current mobile IP specifications [Per96b] allow multiple foreign agents on a network. According to RFC 2002, a mobile detects movement when it no longer receives advertisements from the agent it is registered to. In such a case, it solicits services from any other agent possible and attempts to register through it. Thus, if a foreign agent fails, the mobile can itself recover from the failure, and register through another agent on the same network.

A simple approach to having multiple foreign agents would be to use the above existing mechanism. It is transparent because a mobile detects movement rather than an agent failure. However, we used the home agent redundancy mechanisms described earlier to deal with multiple foreign agents as well. By doing so, we get rid of the numerous Registration messages which would flow across the Internet when all mobiles re-register with their home agents after their foreign agent fails. This allows our approach to have faster recovery from foreign agent failure. Our approach recovers in time proportional to the interval between advertisements. The existing approach, however, requires extra time for mobile nodes to re-register through the new foreign agent across the Internet.

2.4. Binding Synchronization

Our redundancy mechanisms rely on distributing mobility bindings among agent peers. When a mobile M registers with its home agent $HA1$ from a foreign network (Figure 2(a)), $HA1$ replicates its binding on $HA2$ such that $HA2$ can take over from $HA1$ when $HA1$ fails. However, when $HA1$ comes up after a crash (Figure 3), it needs to recover all bindings accumulated on its behalf by $HA2$ when it was down. This looks like a database maintenance problem which could be hard. However, we chose to use simple existing mechanisms.

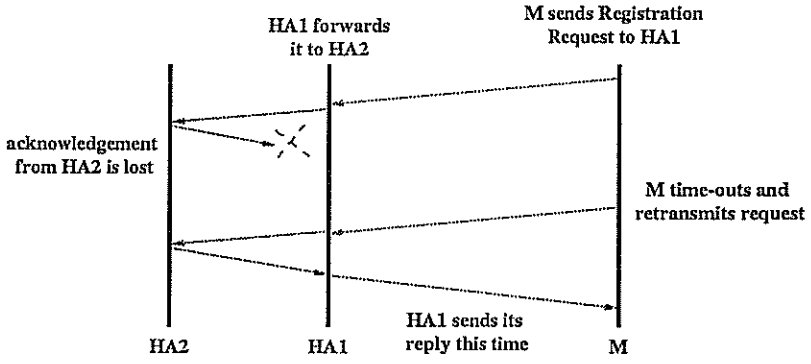


Figure 4: A Time-Space diagram of Home Agent Registration Forwarding; the illustration shows what happens when an acknowledgement is lost. We can also have local retransmission by $HA1$ to handle loss.

Registration Forwarding:. In Figure 2(a), when $HA1$ receives a Registration Request from M, HA first processes it as usual. If it decides to deny service to M, it sends a refusal as before. If it wants to accept the request, it forwards the request to its peer $HA2$. Meanwhile it also stores the registration status of M. $HA2$ stores the mobility binding of M from the forwarded request and sends an acknowledgement back to $HA1$. On receiving the acknowledgement, $HA1$ sends a reply back to M (see Figure 4).

If the forwarded request or its acknowledgement is lost, M retransmits its Registration Request again. Thus, we piggyback the reliability of distributing mobility bindings on the reliability of the Registration messages. This approach is simple. Also, since registration forwarding is done on the local network, the probability of loss should be quite low. If there are many agents on the same network, an agent must wait for acknowledgements from all of them, before it sends its reply.

If there are many backup agents or the loss rate on the local network is high, it might be worthwhile for the home agent to retransmit the registration requests locally rather than have the mobile do it. However, we did not implement local recovery in our current implementation. The Registration Forward and Acknowledgement messages use UDP (the User Datagram Protocol [Pos80]) just like the normal Registration Request and Reply messages in MIP.

Reliable Transfer of Accumulated Bindings using TCP:. When a home agent comes up after crash (Figure 3), we prefer to use an existing transport protocol [BM97] to reliably bulk transfer the mobility bindings which have been accumulated by its peer. The use of TCP is advantageous because the amount of data might be huge, and TCP takes care of most reliability problems. Notice that there is no hidden recursion here because TCP works over fixed IP routing.

A popular inter-domain routing protocol, BGP [RL95], uses the same idea of using TCP for reliable exchange of routing information[Rek95]. Alternatively, we can could also use a low level protocol like BootP[CG85] to do this transfer. However, adhering to our goal of *Simplicity*, we decided to use TCP instead. The home agent must assert itself by ARP prior to the use of TCP. (Otherwise TCP will not work because the peer could have an ARP table that maps the IP address of the recovering agent to its own hardware address!) The mobile can experience a brief period of discontinuity during this interval.

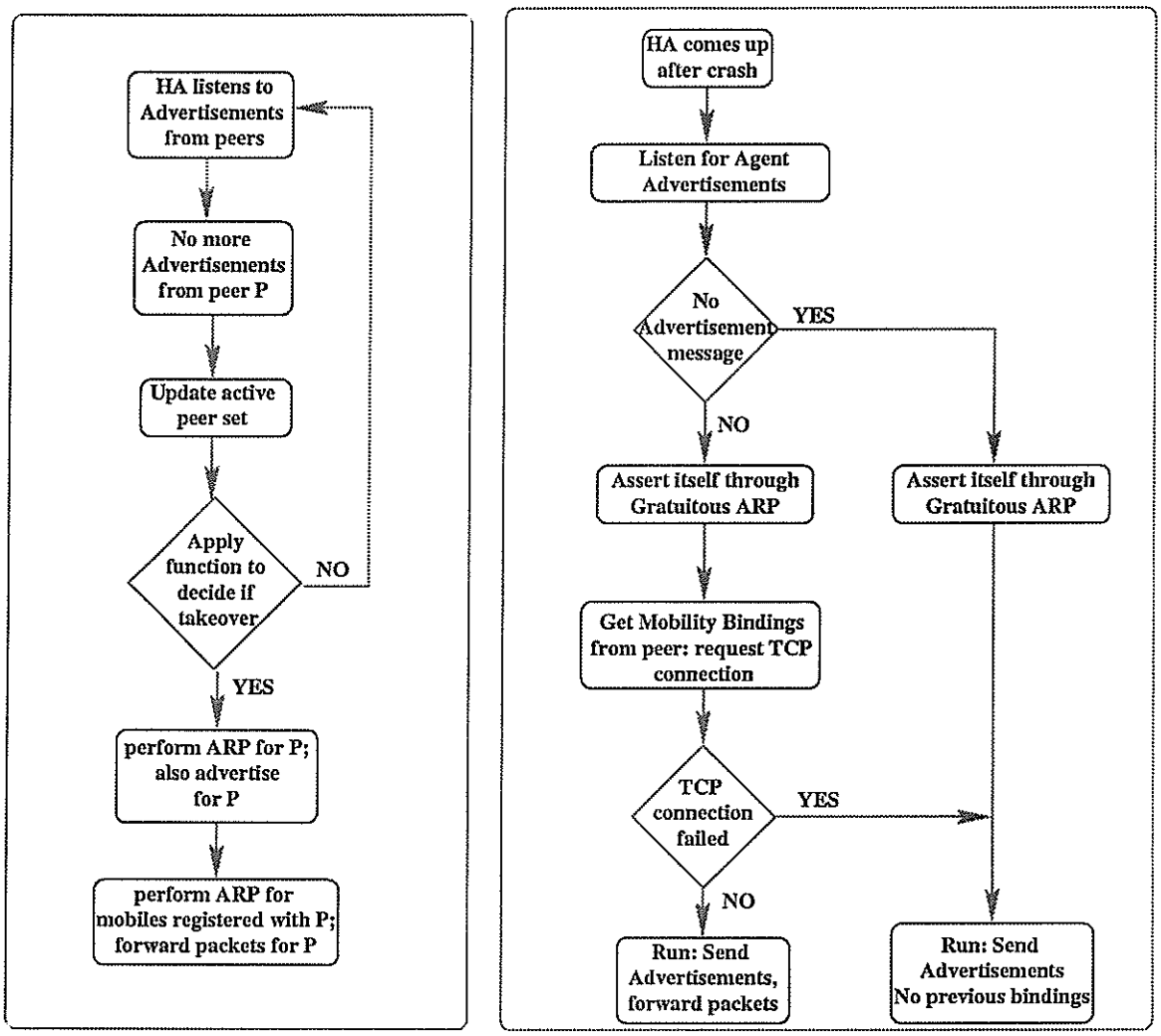
3. Protocol Description

Our FT Mobile IP protocol allows multiple mobility agents on a single network. As of now, our implementation (see Section 4 later) consists of two agents on a LAN. However, we describe our complete design here. Later, we discuss the simple case when there are only two agents on a network.

3.1. Home Agent Considerations

We describe the home agent protocol by first describing the way peers monitor each other and takeover; next, we describe recovery, and finally we describe registration forwarding.

3.1.1. Peer Monitoring and Takeover. Figure 5 contains two flow diagrams that describe the home agent takeover and recovery algorithms respectively. A home agent is statically configured



(a) Home Agent Takeover

(b) Home Agent Recovery

Figure 5: Flow diagrams for the Home Agent Takeover and Recovery algorithms. (a) HA maintains an *Active Peer Set* and applies a boolean function to decide if it should take over. Active Peer Set is updated when HA detects one of its peer to have failed (from absence of Agent Advertisements). (b) HA listens for Advertisements containing its own address. It then obtains its bindings from its peer who was advertising for it. If the TCP connection fails, HA starts with whatever bindings it has received so far.

with the names and addresses of its peers (Peer Set).[2] The aome agent listens to the Agent Advertisements sent out by other aome agents in its *Peer Set* and resets a `takeover_timer` everytime it hears an advertisement.

```
HomeAgentMonitor()    /* Monitor Peers */

  1. Wait for Agent Advertisements from peers.
  2. Received an Agent Advertisement from peer P;
     takeover_timer[P] < – MAX_TAKEOVER_TIMER.
  3. Goto 1.

ActivePeerSet : A list of peers which are alive.

TimerHandler()    /* Called every second */

  1. Decrement takeover_timer[P] for all peers P.
  2. If takeover_timer[P] = 0 for some P
     1. ActivePeerSet < – UpdateActivePeerSet(ActivePeerSet, P, FAILED);
     2. If ApplyFn(ActivePeerSet,P) = True
        Takeover(P).
  3. /* Handle Other Timers */
```

Figure 6: Pseudo code for peer monitoring; `ApplyFn()` is a boolean function which decides if an agent should take over. `ActivePeerSet` excludes P but includes the home agent itself.

Each Home Agent maintains a separate timer for each peer in its Peer Set. If the timer expires for an agent P, agent P is assumed to have failed. The value of the timer must be greater than the period at which an agent advertises itself. Agent Advertisements are usually sent out every 1 - 2 seconds. The `takeover_timer` determines how fast an agent can detect if its peer has failed. However, choosing a small value can lead to false alarms; one must account for loss or delay of Advertisement messages. In our implementation we fixed the `takeover_timer` to be thrice the period at which advertisements were sent. Thus, it would take three consecutive advertisements to be lost before an agent is assumed dead. Figure 6 gives the pseudo code for this.

A home agent maintains an *Active Peer Set* which includes the home agent itself and all its peers which are alive. Upon failure of a peer, the Active Peer Set is updated and a boolean function is applied to it to decide if the home agent should take over. The function should return *True* for one and only one home agent. Consider a scenario where two home agents P and Q fail simultaneously (or, if P was substituting for Q and P fails); `ApplyFn` (Figure 6) could be used to determine which of the remaining home agents take over and which failed home agents they takeover. In such a case, `ApplyFn` could take the failed agent as an argument in addition to the Active Peer Set.

If a home agent HA decides to take over its failed peer P, it performs a Gratuitous ARP (see Figure 5(a)) binding P's network address to its own hardware address. HA must also respond to any ARP requests sent for P. This ensures that it will receive all packets for P transparently (i.e., without any other host on the network being aware of the substitution). HA also performs a similar gratuitous ARP for all the mobiles which were registered with P and are away from home. It also creates IP tunnels [Sim95, Per96a] to their care-of addresses. HA knows the bindings of mobiles that are away from home by virtue of Registration Forwarding (see Section 3.1.3 later). Thus, it can intercept packets for mobiles away from home and tunnel them to their care-of addresses.

[2] We could lift this restriction for static configuration but do not do so here.

```
ProcessRgistrationRequest()

1. if (HomeAgent = Myself)    /* Acting as a Home Agent */
   1. Process request;
   2. if Refuse
      SendRefusal();
   3. else
      1. Forward request to peers;
      2. Receive acknowledgements from peers ;
      3. if all acks received
         SendAcceptance();
2. else if (HomeAgent is my peer and peer is dead and I am taking over)
   /* Acting as a Substitute Home Agent */
   ...
   /* process request on behalf of peer */
   /* Forward to remaining agents in Peer Set and then send Reply */
3. else
   /* False Request */
```

Figure 7: Pseudo code for processing Registration Requests; a home agent must also process requests on behalf of its peer.

All future Registration Requests to P are received by HA; it must recognize them and process them accordingly (see Figure 7). All fresh registration requests are accepted on behalf of P. Registration renewals and de-registration are accepted too (HA has the mobility bindings of P). When replying to these requests, HA writes the address of P in the *Home Agent* field of the reply (Figure 10(b)) so that the mobile host or the foreign agent does not notice any difference. Registrations are also forwarded to the remaining peers of HA. A Reply is sent after receiving Acks from all other members of the Active Peer Set.

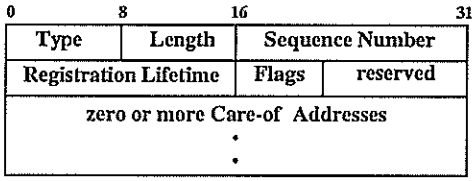
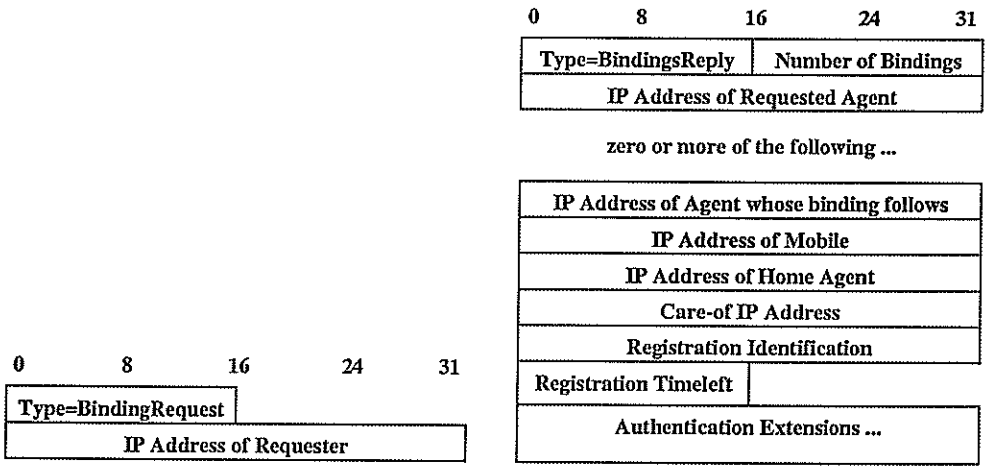


Figure 8: Agent Advertisement Message Format; an agent can advertise multiple care-of addresses.

When a home agent HA takes over its peer P, it must also advertise for P. According to RFC 2002, an Agent Advertisement (Figure 8) can contain multiple care-of addresses. A Home Agent might have to takeover multiple peers (for example, if P was substituting for Q and P fails.) Thus, HA lists all such peers in its agent advertisements. A mobile determines it is at home if it receives an Advertisement containing its home agent's address.

3.1.2. Home Agent Recovery. Figure 5(b) shows the flow diagram for the home agent recovery algorithm. When an agent HA comes up after a crash, it first listens for Agent Advertisements containing its address. If it finds none, it starts functioning afresh (that is, with no previous mobility bindings). Suppose peer P has been advertising for HA — that is P had taken over HA when HA had failed. HA opens a TCP connection to P and requests the mobility bindings accumulated by



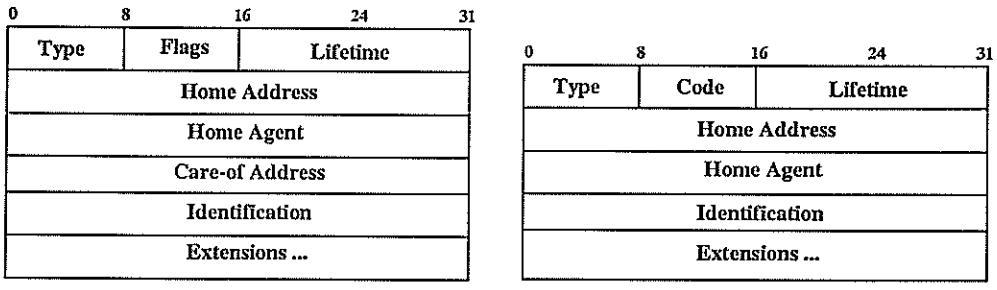
(a) Mobility Binding Request

(b) Mobility Binding Reply

Figure 9: Format of Mobility Binding Request and Reply messages; a Mobility Binding Reply consists of multiple *Mobility Binding Records* which must include the address of the home agent to which the binding actually belongs.

P on behalf of HA. Actually, HA also requires the mobility bindings accumulated by P on behalf of itself and other members of the peer set. This is because P (and other members of the peer set) may fail after HA comes up, and HA may have to take over for them.

Figure 9 shows the format of the Binding Request and Reply messages. A Mobility Binding Reply contains zero or more *Mobility Binding Records*. A Binding Record must include the address of the agent to which the binding actually belongs. The other fields are those present in Registration messages. If all the home agents happen to crash and come up together, they will not receive any advertisement messages containing their addresses. In this case they all start functioning with no mobility bindings. If the TCP connection fails or breaks in the middle, HA resumes with whatever bindings it has received so far.



(a) Registration Request

(b) Registration Reply

Figure 10: Registration Message Formats

3.1.3. Registration Forwarding. A home agent forwards all its registrations to its peers so that if it crashes, one of them can substitute for it. When it receives a registration request (Figure 10(a)) from a mobile host, it first processes it as usual. If it decides to deny service, it sends a refusal (as in ordinary MIP). If it wants to accept the request, it forwards the request to all its peers, and waits for the acknowledgements from all of them. (see the pseudo code in Figure 7). Meanwhile, it also stores the registration bindings of the mobile host.

Type	5 (Registration Request Forwarded)
Flags	same as in original request
Lifetime	- do -
Home Address	- do -
Home Agent	- do -
Care-of Address	- do -
Identification	- do -
Extensions	- do -

Figure 11: Registration Request Forwarded Message; all fields are same as in original request.

The Registration Forward message (Figure 11) has the same fields as in a request(Figure 10(a)). All fields have the same value as in the original request. The purpose of a Registration Forward message is primarily to distribute the registration bindings among the home agents' peers. We choose to use a slight variation of the Registration Request message to keep our protocol simple.

On receipt of a Registration Forward message, the peer agents send back acknowledgements. But, before they do so, they have to process the forwarded request and store the registration bindings. The registration bindings include the mobile's address, its home agents' address, its care-of address, registration lifetime, and identification. Authentication information is recorded too. The peer agents also add a field containing the address of the home agent this mobility binding belongs to. Figure 9(b) lists the fields in a *Mobility Binding Record.*

The Registration Forward Acknowledgement (Figure 12) has the same message format as a Registration Reply (Figure 10(b)). Once again we decided to use the existing Registration Reply format as an Acknowledgement rather than inventing a new one. The *Lifetime* field indicates the minimum of the time requested and the time the agent is willing to serve. All other fields are as in the Forwarded Request. The value of the *Code* field is ignored by the receiving home agent. Peer agents are not allowed to refuse backup services.

Type	7 (Registration Forward Acknowledgement)
Code	0 (ignored by receiver)
Lifetime	minimum of lifetime in request and the time the peer agent is willing to serve.
Home Address	same as in original request
Home Agent	- do -
Care-of Address	- do -
Identification	- do -
Extensions	- do -

Figure 12: Registration Forward Acknowledgement Message

The home agent waits for acknowledgements from all its peers. When it has received Registration

Forward Acknowledgements from all of them, it sends a Reply ((Figure 10(b)) back to the mobile's care-of address accepting its request. The type field is set to 5 (Registration Reply), the code is 1 if the agent does not support simultaneous mobility bindings, else 0. The *Lifetime* field is the minimum of the *Lifetime* fields among all the acknowledgements received and the time the home agent is itself willing to serve. The reason for doing this is to ensure that the mobile will re-register before any of the agents in the peer set timeout the registration. (Ideally, the agents in the peer set should use similar values for *Lifetimes* and other timer fields.) We do not synchronize the deletion of a registration due to timeout; each agent in the peer set times out registrations independently.

The other fields in the Reply are as in the acknowledgement received. If the forwarded registration request or its acknowledgement is lost in the network, it is taken care of by the retransmission of the registration request by the mobile (discussed in Section 2.4). If there are many peers or the network error rate is high, it may be worthwhile for the home agent to retransmit the Registration Request Forwarded message.

3.2. Foreign Agent Considerations

Multiple foreign agents can be handled in the way existing mobile IP does, or it can be handled by our home agent redundancy mechanisms. We decided to use our home agent mechanisms because they are efficient (see the discussion in Section 2.3). The behavior of foreign agents is very similar to that of home agents except for some minor differences.

3.2.1. Peer Monitoring and Takeover. A foreign agent monitors its peer as does a home agent (Section 3.1.1). When a peer P fails, the foreign agent FA performs Gratuitous ARP binding P's network address to its own hardware address. Thus it can receive packets for mobiles registered with P, and decapsulate and forward them.

All future Registration Requests and Replies for P are received by FA. FA accepts all valid Registration Requests on behalf of P. Figure 13 gives the pseudo code for processing Registration Requests. Unlike home agents, foreign agents must forward registration bindings to peers only on receiving an acceptance from the home agent. FA also advertises for P in its Agent Advertisement message.

3.2.2. Foreign Agent Recovery. When a foreign agent FA comes up after a crash, it listens for an Agent Advertisement from one its peers that contains FA's own IP address. As in the case of a home agent, FA requests the mobility bindings from its peer (who was substituting for FA), and then starts functioning. The recovery protocol for a foreign agent is identical to the recovery protocol for a home agent (Section 3.1.2).

3.2.3. Registration Forwarding. Unlike a home agent, a foreign agent forwards its Registration Requests to its peers only when it is about to accept it; that is, after receiving an acceptance from the home agent. Thus a foreign agent will have to generate the original registration request to send to its peers. (The original request contains more information than the Reply; thus is is not sufficient to only relay the Reply to the other agents in the Peer Set.) The foreign agent can do this because it keeps state regarding the mobility bindings for the mobile host while awaiting a reply from the Home Agent. This in turn means that the foreign agent has to store the reply received from the home agent while it is waiting for the acks from all its peers. This stored reply is then sent to the mobile when all acks are received.

```
ProcessRegistrationRequest()

1. if (Care-Of Address = Myself)    /* Acting as a foreign agent */
   1. process request;
   2. if Refuse
      SendRefusal();
   3. else
      1. Forward request to home agent;
      2. Wait for reply from home agent;
      3. If reply is a refusal
         send reply to mobile;
      4. else
         1. Forward original request to all peer agents;
         2. Wait for Acknowledgements from all peers;
         3. If all acks received
            send acceptance to mobile;
2. else if (Care-of Address is my peer and peer is dead and I am taking over)
   /* Acting as a substitute foreign agent */
   ...
   /* Process request on behalf of peer */
   ...
3. else
   /* False Request */
```

Figure 13: Pseudo code for processing a Registration Request at a foreign agent; a foreign agent must also process requests on behalf of its peer.

Figure 13 also gives the pseudo code for forwarding registration requests. Use of the registration forwarding mechanism at both foreign and home agents implies potentially added latency for registration (because of the time waiting for acks from peers), and a slightly increased probability of registration retransmission (because of the extra messages sent to peers whose loss will lead to registration retransmission). We believe these disadvantages are minor and are outweighed by the simplicity of the registration forwarding scheme.

3.3. Mobile Host Considerations

We have tried to change the protocol at the mobile host as little as possible. The only change we made from RFC 2002 [Per96b] concerns how the mobile host does move detection, and determines if it has returned home. As stated earlier (Section 2.2), a mobility agent advertises its peers' addresses if it is substituting for them. However, the first address in an Agent Advertisement message (Figure 8) is the agent's own address.

When a mobile is searching for an agent, it picks up the first address in the agent advertisement it receives. However, to detect movement, it looks at all the addresses in the agent advertisements it receives. If at least one of them is the address of the agent it is registered with, it decides that it has not moved, and it sends registration requests as usual. If it uses the second approach (described in [Per96b]), it should not detect any movement because the network prefix remains the same for both the foreign agents (they are on the same network). A mobile detects it is at home if one of the addresses in the agent advertisements is its home agent address. The above changes had to be made so that the mobile host does not detect any movement when the agent it is registered with crashes.

4. Implementation

We obtained an implementation of Mobile IP (ver 0.95) for the Linux operating system from SUNY Binghamton [GDL96], and ported the code to our machines running NetBSD (ver 1.2E). The SUNY implementation is based on draft version 16 of IP Mobility Support. Then we enhanced it with our fault-tolerant mechanisms. We are implementing our protocol described above (Section 2 and 3). As of now, our implementation allows only two mobility agents on a single network.

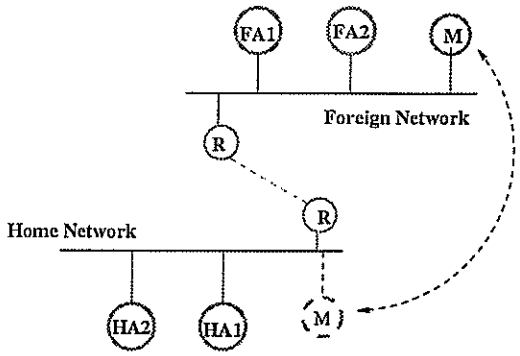


Figure 14: Our Experimental Setup; there are two mobility agents on each network. The mobile M is moved from its home network to a foreign network and vice-versa.

Figure 14 shows our experimental setup. We had two home agents (*HA1* and *HA2*) running on network A and two foreign agents ($FA1$ and $FA2$) running on network B. We moved our mobile M from network A (its home network) to network B and vice-versa. Our protocol is quite simple with only two mobility agent on each network. Agents have only one peer, and so they takeover each other if one fails. `ApplyFn` (see Section 3.1.1) is not required here. Further, when an agent A comes up after a crash, there is no need to wait for Agent Advertisements, since there is only one peer who could have been substituting for A. To verify fault-tolerance, we deliberately crashed one of the home or foreign agents, and later restarted the agent.

We set the value of the `takeover_timer` (see Section 3.1.1) to thrice the period at which agents sent out advertisements (in this case one second). Thus, when one agent failed, its peer was able to react in less than three seconds. Our protocol worked as desired. The mobile observed a brief loss of its packets, but the application running on it handled the packet loss. The process of registration took more time because of registration forwarding to peers. This caused the mobile host to retransmit its request spuriously. To take care of this, we had to increase the value of the retransmit timer in mobile hosts. When a home or foreign agent came up after a crash, the mobile host experienced another brief packet loss.

Figure 15 and 16 show the screen logs of the home and foreign agents processing a Registration Request from a mobile on a foreign network. When a home agent crashed, its peer took over for it. See the screen log in Figure 17.

Our implementation requires a configuration file at the home agent listing the mobile hosts allowed and their authentication information. The implementation we obtained uses the MD5 [Riv92] authentication scheme; we decided to use the same scheme. For our FT enhancements to the implementation, the configuration files at the peer home agents have to be the same; peers share the same mobile - home agent authentication. Foreign agents also have a configuration file listing the visiting mobile hosts allowed and their home agents. The foreign agent peers share the same configuration file; this is because if one agent crashes its peer needs the same information in order to allow the mobile to register.

```
==========================
Sat Jul 26 15:05:41 1997

-- REGISTERME from 128.252.153.70 Port 25094
Id [5240a297:1555d3c0] Type  1 Flags  0 Lifetime      300
Homeaddr: 80fca917, Homeagent: 80fca94a, Careof: 80fc9946
Extension: 32  Length: 16
==========================
Acting as home agent... processing Request

-- FORWARDING REGISTERME to peer 128.252.169.73 Port 45569
.
-- REGFWDACK from peer 128.252.169.73 for MH 80fca917 Code  0

.. Creating tunnel to 128.252.153.70 ..

.. Do ARP for mobile ..
+ arp -s 128.252.169.23 0:60:97:b:1b:b temp pub
+ ifconfig ep0 alias 128.252.169.23

==========================
Sat Jul 26 15:05:42 1997

-- REPLY to 128.252.153.70 at port 25094 (Acceptance)
Id [5061255a:1555d3c0] Type  3 Code   1 Lifetime      300
Homeaddr: 80fca917, Homeagent: 80fca94a
Extension: 32  Length: 16
==========================
```

Figure 15: Screen dump of a home agent (128.252.169.74) processing a Registration Request from a mobile (128.252.169.23) via a foreign agent (128.252.153.70). The home agent accepts the request and forwards it to its peer (128.252.169.73).

Currently, we have not devised any scheme to authenticate between peers. We propose to do one in the future (see Section 5). Our implementation also uses the *Nonces* scheme to protect against replay protection. Thus, when an agent forwards its request to its peer, the peer agent copies and stores the *Identification* field in the registration message so as to be able to accept a request from the mobile directly.

5. Extending other Aspects of Mobile IP

Our protocol so far has concentrated on the basic datagram delivery aspect of mobile IP. In this section we focus on some other aspects like multicasting and we also discuss how we propose to do authentication between peer agents. As of now, we have not implemented the following features.

Routing Multicast Datagrams: As discussed in RFC 2002, multicasting can be handled in two ways. First, a mobile can join the multicast group on the foreign network. This option requires that there be a multicast router present on the visited subnet. If the mobile is using its co-located address, it must use this address as the source address of its IGMP [Dee89] messages; otherwise, it must use its home address. This method is independent of our FT protocol and requires no additional enhancements to our design.

Alternatively, a mobile may join the group via a bi-directional tunnel to its home agent, assuming its home agent is a multicast router. The mobile tunnels IGMP messages to its home agent and the home agent forwards multicast datagrams down the tunnel to the mobile node. This scheme will

```
==========================
Sat Jul 26 15:05:40 1997

-- REGISTERME from 128.252.169.23 Port 10275
Id [5240a297:1555d3c0] Type  1 Flags  0 Lifetime      300
Homeaddr: 80fca917, Homeagent: 80fca94a, Careof: 80fc9946
Extension: 32  Length: 16
==========================
Acting as a foreign agent...

-- REGISTERME FORWARDED TO 128.252.169.74 PORT 45569
..

-- REPLY from HA 80fca94a for VMH 80fca917 Code  1 Lifetime 300

Acceptance from HA
==============================
Processing Accept...

-- FORWARDING REGISTERME to peer 128.252.153.69 Port 45569
.
-- REGFWDACK from peer 128.252.153.69 for MH 80fca917 Code 0

==============================

Sending Acceptance from HA... VMHAddr: 128.252.169.23 Port 10275
Sent Reply to MH...
.......
```

Figure 16: Screen dump of the foreign agent (128.252.153.70) processing a Registration Request from the mobile (128.252.169.23). After receiving the acceptance from the home agent (128.252.169.74), the foreign agent forwards the request to its peer (128.252.153.69).

work in our design as long as the peer agents have the multicast address and they are also multicast routers.

Security and Peer Authentication: Security is a prime concern of any protocol, especially the mobile IP protocol. We do not propose to change any of the security mechanisms described in the MIP specification. However, message passing between mobility agent peers adds another open area of security in our protocol.

Peer agents forward registration bindings among themselves and also request bindings when they come up after a crash. We need a security mechanism to prevent against possible faking and active attacks. Thus, we also need authentication between peers. All messages between peer agents could be followed by the peer authentication extension. The peer authentication could follow the mobile - home agent authentication present in Registration Request Forward and Acknowledgement messages (Figure 11 and 12). The Mobility Binding Request and Reply (Figure 9) should also contain the peer authentication. The default authentication method could be the MD5 [Riv92] authentication scheme used between the mobile and its home agent.

As mentioned earlier, peer home agents should also share the same authentication information with mobile hosts.

```
-- Failure detected on peer 128.252.169.74

- Temporarily publish peers IP address bound to my hardware address
+ arp -s 128.252.169.74 0:60:97:b:1b:b temp pub

- Configure my interface to accept packets for my peer
+ ifconfig ep0 alias 128.252.169.74

- Do ARP for mobile hosts ..
+ arp -s 128.252.169.23 0:60:97:b:1b:b temp pub
+ ifconfig ep0 alias 128.252.169.23

- Creating tunnels to ...
+ 128.252.153.70
```

Figure 17: Log of a home agent taking over its peer; the home agent also performs ARP for the mobile host (128.252.169.23) and creates a tunnel to its care-of address (128.252.153.70).

```
...

Connecting to peer 128.252.169.73 ...
-- REQUEST BINDINGS to 128.252.169.73 Port 12835.

-- BINDINGS REPLY received from 128.252.169.73 ...
Number of bindings = 3

- Binding for 80fca94a (self):
  Id:[3ce02931:2707cc46]  Mobile:80fca917  HomeAgent:80fca94a
  CareOf:       0  Timeleft:0

- Binding for 80fca949 (peer):
  Id:[3329ec06:1112e484]  Mobile:80fca945  HomeAgent:80fca949
  CareOf:80fc9946  Timeleft:220

- Binding for 80fca94a (self):
  Id:[51ad2490:1cd59c2a]  Mobile:80fca919  HomeAgent:80fca94a
  CareOf:80fc9946  Timeleft:150

Closing connection ...
...
```

Figure 18: Screen dump of a home agent (128.252.169.74) obtaining its bindings from its peer (128.252.169.73). The authentication extension is not displayed.

6. Conclusion

As laptops begin to proliferate, it is increasingly important for mobile nodes to be able to have seamless Internet communication as the mobile nodes from network to network. Mobile IP provides an extremely simple and elegant solution to the problem of seamless IP communication by forwarding packets through home and foreign agents. However, the current protocol makes no provision for home agent failure, and for load balancing to prevent agents from becoming bottlenecks. As the Mobile IP protocol begins to be widely deployed, we believe that both fault-tolerance and load balancing will become essential features of commercial offerings.

In this paper, we have described a simple extension of the existing mobile IP protocol to allow multiple redundant agents that can rapidly take over from each other in case of failure. Our protocol also allows load balancing (though one could argue that the existing Mobile IP specification allowed

such load balancing in the first place though it is not explicitly called out). Our design was guided by the desire to be compatible with both the details and the simple design philosophy of existing mobile IP. We also took care to change the mobile host protocol as little as possible to allow our protocol to be easily deployed in existing mobile IP environments.

The only change to existing mobile nodes that we require is for a mobile host at home not to detect a move when its primary home agent H crashes and the peer agent lists H in its advertisements. This is a small change and is only required if the mobile is using home agent addresses in advertisements to detect movement. If we want to avoid even this small change (because it potentially involves reprogramming many existing mobile nodes), two approaches are possible.

First, we can have a peer agent send a separate advertisement for each home agent it has taken over for (rather than send a single advertisement listing all agents it is substituting for). Unfortunately, this might require that the peer spoof the source IP address of the original home agent that has crashed. A second possibility is to allow the mobile to detect a spurious move when its home agent crashes. The mobile will then register with a supposed foreign agent on the same network. As long as this foreign agent is smart enough to recognize that no real move has taken place, the foreign agent will do nothing, and packets will continue to reach the mobile agent at home. Neither of these two possibilities are perfectly clean solutions, but they do potentially allow complete transparency.

We have implemented an initial version of our fault tolerant mobile IP protocol. We have verified the basic working of the protocol by manually crashing and restarting agents and by observing the protocol messages sent and the system log. We are planning to complete our implementation and test the performance of real applications when failures occur, and do some measurements. We plan to make our code publically available on the world-wide web.

Other future work includes a careful incorporation of all remaining features of the existing Mobile IP specification including multicast and security. While each feature in Mobile IP is fairly simple, the combination of features together with fault-tolerance requires some careful verification to ensure that there are no unsuspected interactions.

References

[BM97] James R. Binkley and John McHugh. Secure Mobile Networking: Sixth Quarterly Report - Winter 1997. *Portland State University*, April 1997. Web address http://www.cs.pdx.edu/research/SMN/1q97.ps.

[CG85] Bill Croft and John Gilmore. BOOTSTRAP PROTOCOL (BOOTP). *RFC 951*, September 1985.

[Dee89] S. Deering. Host Extensions for IP Multicasting. *STD 5 RFC 1112*, August 1989.

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

[Dro93] R. Droms. Dynamic Host Configuration Protocol. *RFC 1541*, October 1993.

[GDL96] Vipul Gupta, Abhijit Dixit, and Ben Lancki. Linux Mobile-IP Home Page. *SUNY Binghampton*, May 1996. Web address http://anchor.cs.binghamton.edu/ mobileip/.

[Joh91] John Ioannidis and Dan Duchamp and Gerald Q. Maguire Jr. IP-based Protocols for Mobile Networking. In *Proceedings of the SIGCOMM '91 conference: Communications Architectures and Protocols*, pages 235–245, September 1991.

[Joh94] David B. Johnson. Scalable and Robust Internetwork Routing for Mobile Hosts. In *Proceedings of the 14th International Conference on Distributed Computing Systems*, pages 2–11, june 1994.

[MS93] Andrew Myles and David Skellern. Comparing Four IP Based Mobile Host Protocols. *Computer Networks and ISDN Systems*, pages 349–356, November 1993.

[Per96a] C. Perkins. IP Encapsulation Within IP. *RFC 2003*, October 1996.

[Per96b] C. Perkins. IP Mobility Support. *RFC 2002*, October 1996.

[Per96c] C. Perkins. Minimal Encapsulation with IP. *RFC 2004*, October 1996.

[Plu82] David C. Plummer. An Ethernet Address Resolution Protocol – or – Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware . *STD 37, RFC 826*, November 1982.

[PMJ94] Charles Perkins, Andrew Myles, and David B. Johnson. The Internet Mobile Host Protocol. In *Proceedings of the 2nd International Conference on Universal Wireless Access*, pages 197–202, April 1994.

[Pos80] J. Postel. User Datagram Protocol. *STD 6, RFC 768*, August 1980.

[Pos81] J. Postel. Internet Protocol. *STD5, RFC 791*, September 1981.

[Rek95] Yakov Rekhter. Inter-Domain Routing: EGP, BGP, and IDRP. *Routing in Communications Networks by Martha Steenstrup*, pages 99–133, chapter 4, Prentice Hall, 1995.

[Riv92] R. Rivest. The MD5 Message-Digest Algorithm. *RFC 1321*, April 1992.

[RL95] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). *RFC 1771*, March 1995.

[Sim95] W. Simpson. IP in IP Tunneling. *RFC 1853*, October 1995.

[Ste94] W. Richard Stevens. *TCP/IP Illustrated, Volume 1, The Protocols.* Addison-Wesley Professional Computing Series, 1994.

[TCT92] F. Terroka, Kim Claffy, and M. Tokoro. Design, Implementation, and Evaluation of Virtual Internet Protocol. In *Proceedings of the 12th International Conference on Distributed Computing Systems*, pages 170–177, June 1992.

[WS94] Gary R. Wright and W. Richard Stevens. *TCP/IP Illustrated, Volume 2, The Implementation.* Addison-Wesley Professional Computing Series, 1994.