Mobile-IP for Linux (ver 1.00)

Abhijit Dixit and Vipul Gupta
Dept. of Computer Science
State University of New York
Binghamton, NY 13902
{*abhijit,vgupta*}*@cs.binghamton.edu*

May 23, 1996

Abstract

This release note describes version 1.00 of Linux Mobile-IP developed at the State University of New York, Binghamton. Except as noted, this software complies with revision 16 of the IETF Mobile-IP draft. It improves upon two prior releases (versions 0.90 and 0.95) and even allows a mobile host to operate in the absence of foreign agents.

The software runs in user space but requires IPIP tunneling support which is available in Linux kernels after 1.3 (we use 1.3.58). This article includes a brief introduction to Mobile-IP, an overview of our implementation, a user guide and contact information for the authors.

1 Introduction

Recent advances in hardware and communication technologies have ushered in the era of mobile computing. The proliferation of powerful notebook computers and wireless communications promises to provide users with network access at all times. This continuous connectivity will allow users to be quickly notified of changing events and provide them with the resources necessary to respond to them even when in transit. However, present day internetworking protocols such as IP (also IPX and Appletalk) behave awkwardly when dealing with host migration from one network to another.

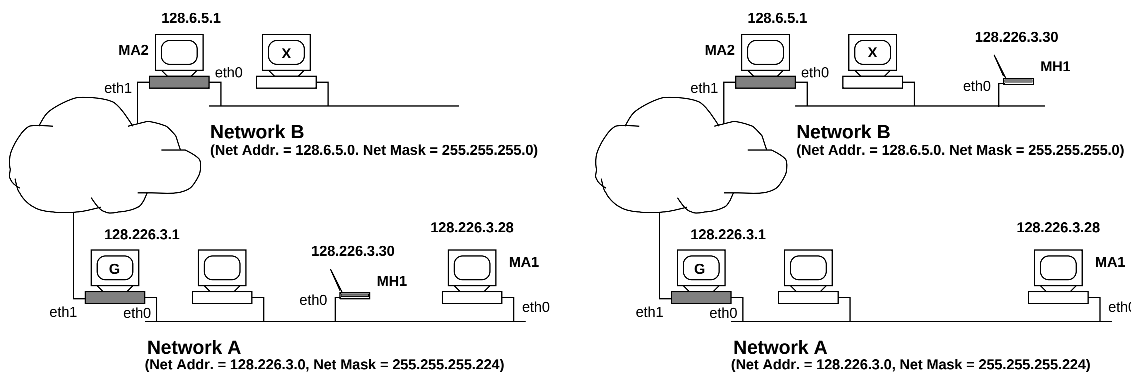
Current versions of the Internet Protocol (IP) make an implicit assumption that the point at which a computer attaches to the Internet is fixed and its IP address identifies the network to which it is attached. Datagrams are sent to a computer based on the location information contained in its IP address. If a computer (a.k.a *host*) moves to a new network while keeping its IP address unchanged, its address will not reflect its new point of attachment. Consequently, existing routing protocols will be unable to route datagrams to it correctly. In this situation, the mobile node (a.k.a. mobile host) must be reconfigured with a different IP address representative of its new location. Not only is this process cumbersome for ordinary users, it also presents the problem of informing potential correspondents of the new address. Furthermore, changing the IP address will cause already established transport layer connections to be lost. Put simply, if the mobile host moves without changing its address, it will lose routing; and if it does change its address, it will lose connections. Mobile-IP is an enhancement to IP which allows a computer to roam freely on the Internet while being reachable at the same IP address.

The Internet Engineering Task Force (IETF) [7] is already working on defining a Mobile-IP standard [11]. Except as noted in Section 6, our software is fully compliant with the 16th revision of this proposed standard.

The rest of this article is organized into five sections. Section 2 is a brief overview of Mobile-IP. Section 3 presents specific details of our implementation of Mobile-IP. It also discusses an example internetwork with mobility support which serves to illustrate many of the ideas expressed in later sections. Section 4 contains a user guide for this software. Section 5 lists the major changes in this software relative its previous version. Section 6 discusses current limitations and other issues that users should be aware of. Finally, Section 7 includes acknowledgments and contact information for the authors.

2 Overview of Mobile-IP

The IETF Mobile-IP architecture defines special entities called the Home Agent (HA) and Foreign Agent (FA) which cooperate to allow a Mobile Host (MH) to move without changing its IP address. The term *mobility agent* is used to refer to a host acting either as a HA or FA (or both). We describe a network as *mobility supporting* if it is equipped with a mobility agent.



(a) (b)

Figure 1: A wired internetwork with mobility support. In (a), the mobile host MH1 is home. In (b), MH1 is visiting Network B.

Each MH is associated with a unique *home network* as indicated by its permanent IP address. Normal IP routing always delivers packets meant for the MH to this network. When a MH is away, a specially designated host (the Home Agent) on this network is responsible for intercepting and forwarding its packets. The MH uses a special registration protocol to keep its HA informed about its current location. Whenever a MH moves from its home network to a foreign network, or from one foreign network to another, it chooses a Foreign Agent (FA) on the new network and uses it to forward a registration message to its HA. After a successful registration, packets arriving for the MH on its home network are *encapsulated* by its HA and sent to its FA. Encapsulation refers to the process of enclosing the original datagram as data inside another datagram with a new IP header.[1] The source and destination addresses fields in the outer header correspond to the HA

[1] This is similar to the post office affixing a new label over an older address label when forwarding mail for a customer who has moved.

and FA, respectively. This mechanism is also called *tunneling* since intermediate routers remain oblivious of the inner (original) IP header.[2] On receiving the encapsulated datagram, the FA strips off the outer header (this process is termed *decapsulation*) and delivers the newly exposed datagram to the appropriate visiting MH on its local network.

The IETF architecture also allows a MH to do its own decapsulation. In this case, the MH must acquire a temporary IP address on the foreign network (*e.g.* using DHCP [3] or even PPP) to be used for forwarding. Decapsulation at the mobile host extends the applicability of Mobile-IP to networks that do not provide foreign agents. However, it does require the MH to handle encapsulated packets and to recognize both the newly acquired address and the permanent IP address as its own. Self decapsulation by the MH is a new feature in this version of our software.

3 Linux Mobile-IP

Since Linux [10] runs on PCs (which tend to be cheaper than workstations) and (like our software) is available for free, we feel that Linux Mobile-IP is an economically attractive choice for setting up a mobility supporting internetwork. This section contains a high-level description of our implementation which should be of interest to other implementors of this protocol. For now, we assume that a foreign agent is available on all networks which a MH visits. In a later section, we consider the case when this assumption does not hold. Figure 1 shows a mobility supporting internetwork which serves as an illustrative example for the subsequent discussion. It shows two mobility supporting networks, Network A and Network B, which are equipped with mobility agents MA1 and MA2, respectively. A mobile host, MH1, is also shown for which the home network is Network A. Whenever MH1 is away, MA1 acts as its home agent. When MH1 visits Network B, MA2 acts as its foreign agent.

While this example illustrates a wired internetwork, our software works just as effectively in a wireless environment. It includes two daemon programs *mh* and *agent* that incorporate the appropriate behavior for a mobile host and a mobility agent, respectively. Both programs run in user space rather than being compiled into the kernel. Since these daemons need the ability to manipulate kernel routing tables and network interfaces, they must be run with root permissions. Both the *mh* and *agent* programs rely on kernel features that are Linux specific but at least the *mh* should be easily portable to other UNIX variants especially if self decapsulation is disabled. In order to use our software, the Linux kernel on all mobility-aware hosts should be configured with support for the `/proc` file system, IP multicasting, IPIP (IP-within-IP) encapsulation, IP forwarding, and appropriate networking cards. The mobile hosts also need to include the *dummy* network interface to operate successfully on foreign networks that are not equipped with foreign agents. It is possible to use IP aliasing instead of the dummy interface but doing so will require changes to our code. Note that the IETF Mobile-IP protocol is not designed to work with firewalls so IP firewalling should not be enabled.

3.1 Implementation Overview

Implementation details of Linux Mobile-IP are best explained by separately considering (a) the routing table changes that must be made in response to host migration, and (b) the scheduling of

[2] In the absence of this encapsulation, intermediate routers will simply return packets back to the home network.

these changes. These changes are made automatically by the *mh* and *agent* daemons without any user intervention. For the example internetwork shown in Figure 1, migration of MH1 between Network A and Network B will only affect the routing table entries on MA1 (home agent), MA2 (foreign agent) and MH1 (mobile host). When the mobile node is home, normal IP routing is sufficient to correctly route packets to and from it. In this case, the routing tables on MA1, MA2 and MH1 are created by the network initialization commands shown in Figure 2.

When MH1 moves to Network B, it must change its default route to go through MA2 and delete routes that are no longer reachable through interface *eth0*. The foreign agent, MA2, must set a host specific route for MH1 through *eth0*. On Network A, MA1 must arrange to intercept MH1's packets and forward them to MA2 after encapsulation. These actions are equivalent to executing the commands shown in Figure 3. The home agent, MA1, establishes a proxy-ARP entry for MH1 and sets up a host specific route to it through a special software interface called a *tunnel* [9]. The tunnel interface is available in newer Linux kernels (1.3.x) and can be used for IPIP encapsulation. When a tunnel is configured as a point-to-point interface with IP addresses P and Q, any packets received by it are enclosed within a new IP header before being re-routed. The protocol field in this header is set to IPIP and P and Q are used as the source and destination addresses, respectively. The Linux kernel automatically decapsulates all IPIP packets and re-routes the newly exposed datagram. For a home agent to support n different mobile hosts, its kernel must have a sufficient number of tunnel interfaces *tunl1*, *tunl2*, ... , *tunln*. More of these interfaces can either be dynamically loaded or compiled into the kernel by modifying `/usr/src/linux/drivers/net/Space.c` [5].

Before the necessary changes can be made to the routing tables, the mobile host must figure out its current location and relay this information to its home agent. Each mobility agent periodically broadcasts a special agent advertisement message on the mobility supporting network. These are extensions of ICMP router discovery messages [2] and are sent to the multicast address `224.0.0.1`. A mobile host listens for these messages and deduces its current location from their content. If a mobile host starts receiving agent advertisements from multiple mobility agents (as may happen if wireless cells overlap), it retains the service of its previous mobility agent till that agent's advertisements are no longer heard. This prevents oscillatory behavior where a mobile host may rapidly switch back and forth between multiple agents. A better approach would be to consider the advertisement's signal strength when making this switch. However, this requires new hardware-specific `ioctl()` calls to be added to the Linux kernel [4].

A mobile host uses the four-step registration shown in Figure 4 to inform its home agent of its current location. The registration request sent by the mobile host includes the care-of address which its home agent must use for forwarding datagrams and is authenticated to prevent misuse. Registration may fail due to a variety of reasons including authentication failure, non-availability of resources (e.g. tunnel interfaces) or due to the administrative policy in effect. When a mobile host returns home, it undergoes a deregistration process which appropriately resets its routing tables and those on its home agent. The (de)registration process uses an unreliable protocol (UDP) and the mobile host is responsible for retransmitting lost messages. Additionally, mobility agents use a status log, written to non-volatile storage, to recover quickly from planned reboots.

Once the *mh* daemon has been started on MH1, and the *agent* daemons on MA1 and MA2; MH1 can move freely between Network A and Network B and still communicate with the rest of the Internet using the same IP address. When MH1 is away from both networks, applications (*e.g.* rlogin, telnet, ftp, Netscape) involving MH1 will experience a momentary loss in connectivity.

```
On MH1
  # Initialize the loopback and eth0 interfaces
  ifconfig lo 127.0.0.1
  ifconfig eth0 128.226.3.30 netmask 255.255.255.224 broadcast 128.226.3.63
  # Set up routing tables
  route add -net 127.0.0.0
  route add -host 128.226.3.28 dev eth0
  # Note that MH1 uses MA1 rather than G as
  # its default router
  route add default gw 128.226.3.28 metric 1

On MA1:
  # Initialize interfaces
  ifconfig lo 127.0.0.1
  ifconfig eth0 128.226.3.28 netmask 255.255.255.224 broadcast 128.226.3.63
  # Initialize routing table
  route add -net 127.0.0.0
  route add -net 128.226.3.0 netmask 255.255.255.224 dev eth0
  route add default gw 128.226.3.1 metric 1

On MA2:
  # Initialize interfaces (p.q.r.s is the IP address for eth1)
  ifconfig lo 127.0.0.1
  ifconfig eth0 128.6.5.1 netmask 255.255.255.0 broadcast 128.6.5.255
  ifconfig eth1 p.q.r.s       ...
  # Initialize routing table (u.v.w.x is the default gateway
  # on the same network as eth1)
  route add -net 127.0.0.0
  route add -net 128.6.5.0 netmask 255.255.255.0 dev eth0
  route add default gw u.v.w.x metric 1
```

Figure 2: Contents of `/etc/rc.d/rc.inet1` (or equivalent) on the mobility agents and mobile host

```
On MH1
  # Delete the route previously reachable through eth0
  route del 128.226.3.28
  # Make MA2 the new default router.
  route add -host 128.6.5.1 dev eth0
  route add default gw 128.6.5.1 metric 1

On MA1:
  # Initialize a tunnel interface
  ifconfig tunl1 128.226.3.28 netmask 255.255.255.224 pointopoint 128.6.5.1
  # Set a route to MH1 through the tunnel interface
  route add -host 128.226.3.30 dev tunl1
  # Arrange to intercept packets meant for MH1 by
  # advertising our own hardware address as that of MH1
  /sbin/arp -s 128.226.3.30 08:1E:19:07:2C:53 pub

On MA2:
  # Set up host specific route to visiting mobile host
  route add -host 128.226.3.30 dev eth0
```

Figure 3: Routing changes required when MH1 moves to Network B.

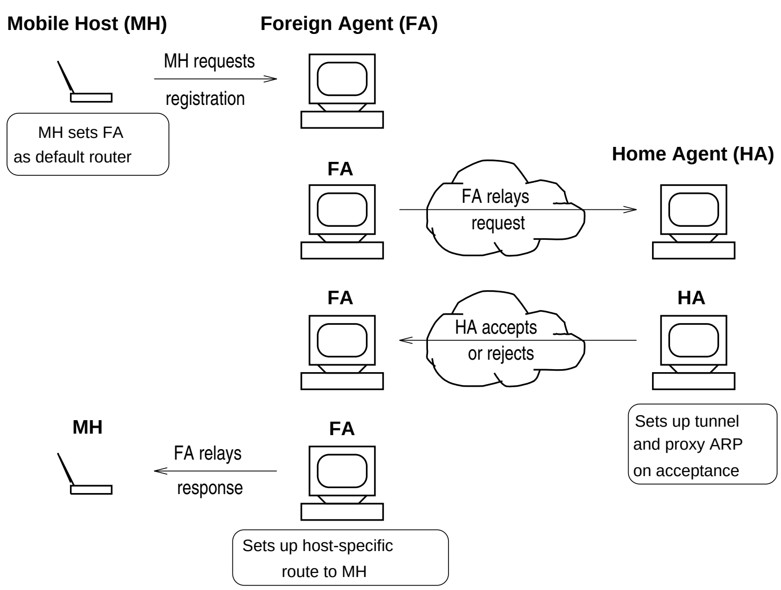


Figure 4: Four phases in the registration process.

That connectivity will be transparently restored soon after MH1 rejoins either network.

3.2 Coping without Foreign Agents

As mentioned in Section 2, allowing self-decapsulation at the mobile host obviates the presence of a foreign agent. Figure 5 is similar to Figure 1(b) except that there is no foreign agent on Network B (*i.e.* G2 is either incapable or has chosen not to act as a foreign agent). Suppose that MH1 has temporarily acquired the IP address 128.6.5.3 on Network B. The mobile host must now accept packets sent to both 128.226.3.30 and 128.6.5.3. Linux already provides a *dummy* interface to force a host to treat multiple IP addresses as its own. The HA should set up its routing tables to encapsulate packets meant for 128.226.3.30 and forward them to 128.6.5.3. Figure 6 indicates the equivalent commands that must be executed on HA and MA1 to accomplish these changes. The overall effect is to set up a tunnel from MA1 to MH1 and assign MH1's permanent IP address to *dummy* and the temporary address to *eth0*.

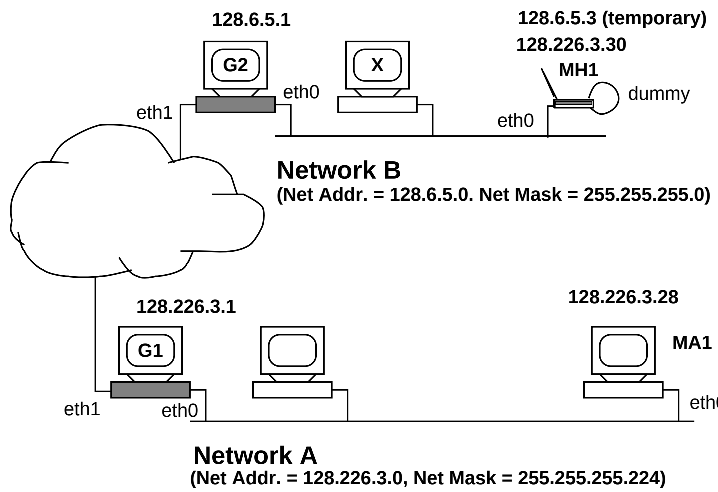


Figure 5: An example illustrating mobility support without a foreign agent.

While it may not be obvious at first, this arrangement does pose a problem for connections initiated by a mobile host while on a foreign network. At the time of connection initiation, the source address used for the connection is the temporarily leased IP address (as that is the address assigned to *eth0*). Since this address is no longer associated with the mobile host when it moves to another network, connections initiated by the mobile host while on a foreign network will not survive. There is a simple solution to this problem which changes the behavior of the mobile host only slightly. Instead of assigning the temporary IP address to *eth0*, we assign that address to the *dummy* interface (the permanent address is assigned to *eth0*) and use proxy-ARP to pick up packets for the temporary address on the foreign network. This corresponds to replacing the commands shown for MH1 in Figure 6 by those shown in Figure 7. With this change, connections initiated by a mobile host always use its permanent IP address as the source.

The *mh* program continuously monitors the IP address on the network interface through which its default route passes. If the IP address on this interface changes to something other than the mobile host's permanent IP address and no foreign agents can be 'heard', the *mh* program puts the mobile host in the 'self-decapsulation' mode. This change may occur, for example, when a user replaces the PCMCIA network card on her laptop with a modem card and initiates PPP.

```
On MA1
  # Set up a tunnel termination at the temp IP address
  ifconfig tunl1 128.226.3.28 netmask 255.255.255.224 pointopoint 128.6.5.3
  # Set a route to MH1 through the tunnel interface
  route add -host 128.226.3.30 dev tunl1
  # Arrange to intercept packets meant for MH1 by
  # advertising our own HWaddress as that of MH1
  /sbin/arp -s 128.226.3.30 08:1E:19:07:2C:53 pub

On MH1
  # Assign the newly acquired address to eth0
  ifconfig eth0 128.6.5.3
  # Set G2 as the default router
  route add -host 128.6.5.1 dev eth0
  route add default gw 128.6.5.1 metric 1
  # Delete the route previously reachable through eth0
  route del 128.226.3.28
  # Assign the permanent IP address to a `place-holder´
  # interface so packets meant for that address are
  # recognized as our own
  ifconfig dummy 128.226.3.30
  route add 128.226.3.30 dev dummy
```

Figure 6: Routing changes required to support mobility without foreign agents (Case I).

4 User Guide

4.1 Compiling the daemons

The *agent* daemon is created by compiling agent.c, fileio.c, low.c, auth.c and util.c using a command similar to

```
gcc -o agent agent.c fileio.c low.c auth.c util.c
```

The *mh* daemon is created similarly by compiling mh.c, mhfileio.c, mhlow.c, auth.c and util.c. A Makefile is included with the software distribution and simply typing `make` should correctly compile both daemons. Alternatively, one may use `make agent` or `make mh` for selective compilation.

4.2 Running the agent daemon

The host which serves as the home agent should have a file named `/etc/mip-mh.ok`. The file contains information about each mobile host supported by the home agent. For the test bed shown

```
On MH1
  # Choose G2 as default router
  route add -host 128.6.5.1 dev eth0
  route add default gw 128.6.5.1 dev eth0
  # Delete the route previously reachable through eth0
  route del 128.226.3.28
  # Retain the permanent address on eth0, assign the
  # temporarily acquired address to dummy and proxy-ARP
  # for that address. Here 00:20:AF:D1:BF:04 is the
  # hardware address for eth1 on MH1
  ifconfig dummy 128.6.5.3
  route add 128.6.5.3 dev dummy
  arp -s 128.6.5.3 00:20:AF:D1:BF:04 pub
```

Figure 7: Routing changes required to support mobility without foreign agents (Case II).

in Figure 1, its contents would be:

```
1
128.226.3.30 197167 md5 16 0f 56 80 34 fe 30 b4 9a 5e 12 ef 5c 9e 67 33 72
```

The first line indicates the number of mobile hosts for which this computer may act as the home agent. If n mobile hosts are supported, there are n lines (one per mobile host) following the first one. From left to right, each line lists the permanent IP address of the mobile host, the SPI value (decimal) used for authentication, the authentication algorithm, the key length in bytes and the key value (with each byte specified as a hexadecimal number). Currently, we only support the default authentication mechanism which is MD5 in the prefix-suffix mode. The SPI value can be arbitrary as long as the mobile host and its home agent agree to use the same one. The key length could also be arbitrary but we use sixteen bytes as that matches the length of the MD5 signature [12]. This key acts as a "shared secret" used for authentication between the mobile host and its home agent.

A machine which acts as a foreign agent (*e.g.* MA2 in Figure 1) should have a file named `/etc/mip-vmh.ok` which contains information about authorized visiting mobile hosts. Currently, only the IP addresses of a mobile host and its home agent are included. Later versions of the software may support authentication between mobile hosts and foreign agents. In that situation, this file would also include authentication information. For Figure 1, this file contains:

```
1
128.226.3.30 128.226.3.28
```

The first line indicates the number of mobile hosts that this foreign agent is willing to support and subsequent lines contain one entry for each authorized visitor.

A mobility agent that intends to act as a home agent for some mobile hosts and as a foreign agent for some others must have both `/etc/mip-mh.ok` and `/etc/mip-vmh.ok`. As a security measure, these files should be readable only by the superuser.

The *agent* daemon requires the following command line arguments:

-a : agent's IP address on the mobility supporting network
-m : network mask corresponding to the IP address above
-i : name of the network interface (*e.g.* eth0) on the mobility supporting network
-h : hardware address of this interface

For the setup shown in Figure 1, the *agent* daemon on MA1 is started as

```
agent -a 128.226.3.28 -m 255.255.255.224 -h 08:1E:19:07:2C:53 -i eth0
```

where `08:1E:19:07:2C:53` is the hardware address of the *eth0* interface. The value passed with the `-h` option is used only if a mobility agent will serve as a home agent for some host. In Figure 1, MA2 acts only as a foreign agent and the *agent* program is started as

```
agent -a 128.6.5.1 -m 255.255.255.0 -h 00:00:00:00:00:00 -i eth0
```

4.3 Running the mh daemon

The *mh* daemon also requires a configuration file which is named `/etc/mip-ha.dat` and contains a single line with authentication information for the mobile host's home agent. For Figure 1, its contents on MH1 are:

```
197167 md5 16 0f 56 80 34 fe 30 b4 9a 5e 12 ef 5c 9e 67 33 72
```

These fields indicate the SPI value, authentication type (currently only md5 is supported), key length in bytes and the key value. The SPI value, key length and key value used here should be the same as that expected by the home agent for this mobile host.

Command line arguments for the *mh* daemon are:

-a : mobile host's permanent IP address
-m : network mask for the home network
-g : home agent's IP address

In Figure 1, the *mh* daemon on MH1 is started as

```
mh -a 128.226.3.30 -m 255.255.225.224 -g 128.226.3.28
```

Files `agent.c` and `mh.c` contain a variable called `debug` whose value can be varied in the range 0–3 to control the verbosity of the daemon programs as they run. Higher values correspond to more verbose messages and the default value is two. These messages are printed on `stderr` and can be saved in a log file using appropriate file redirection.

Once the *mh* daemon has been started on MH1, and the *agent* daemons on MA1 and MA2; MH1 can move freely between Network A and Network B and still communicate with the rest of the Internet using the same IP address. During periods when MH1 is on neither network, network applications (*e.g.* rlogin, telnet, ftp, Netscape) that involve MH1 will experience a momentary loss in connectivity. That connectivity will be transparently restored soon after MH1 rejoins either Network A or Network B.

5 Significant Changes from the Previous Release

Router discovery ICMP messages [2] with appropriate mobility extensions are now used for agent advertisements and solicitations (version 0.95 used proprietary messages). The home agent and the mobile host send out gratuitous ARPs when appropriate and the home agent ensures that rejections to deregistration requests originating on the home network are sent on the home network (even if a tunnel exists for the requesting mobile host). All authentication extensions now include SPI values and the type and length fields are included in computing the authenticator.

The *mh* program has been completely redesigned to allow a mobile host to operate on foreign networks that are not equipped with foreign agents. For example, we are able to move a portable computer back and forth between an ethernet LAN in our lab and a PPP connection at an offcampus location (five miles away) without disturbing any TCP connections. To the best of our knowledge, ours is the first publicly available Mobile-IP implementation for Linux with this feature.

6 Known Limitations and Other Issues

The IETF Mobile-IP draft specifies two kinds of protection mechanisms against replay attacks but our software only supports nonce-based replay protection. The draft requires that a mobile host may not send any broadcast ARP packets nor reply to ARP requests for its address unless they originate at the foreign agent. This behavior is not implemented in this release. Incorporating this feature would have required users to apply a patch to the standard kernel distribution. We are currently in the process of testing other kernel modifications (described below) and decided to defer distribution of all patches until the impending release of Linux 1.4.x, *a.k.a.* Linux 2.0 (the 1.3.x kernels are considered experimental). In many cases, the omission of these ARP related features will not have any adverse effects. Readers interested in the full implications of this omission are referred to the IETF draft and Mobile-IP transparencies available at our web-site (see Section 7).

The following additional limitations of the previous version still persist since fixing them would require kernel patches. None of these is very serious and kernel changes (against Linux 1.3.58) that incorporate proposed solutions have been implemented by students at SUNY Binghamton as projects for a graduate course. These changes have undergone only preliminary testing so far but are available to interested individuals as indicated on our web site. Note, however, that our user-level code would have to be tweaked to exercise the kernel features implemented by these changes.

Even when a mobile host leaves a foreign network, its foreign agent on that network retains a local route for it for some time (equal to the remaining time on its most recent registration renewal). For example, when MH1 moves from Network B to Network A, MA2 retains a host specific route for MH1 through its interface *eth0*. As a result, for hosts such as X whose path to MH1 goes through MA2, MH1 is temporarily unreachable. This problem can be solved by changing the IPIP handler on MA2 to set up and tear down the local route to MH1 on a packet-by-packet basis [6].

The standard Linux kernel does not process gratuitous ARP requests properly. This may result in another situation where a mobile host is temporarily unreachable. Considering a situation in Figure 1 where MH1 has recently moved from Network A to Network B and successfully registered this change with MA1. At this time, MA1 would have set a proxy ARP entry for MH1. However, other computers on Network A that do not process gratuitous ARPs properly may retain a stale ARP entry for MH1 (from the time it was on Network A). These computers will not forward packets meant for MH1 to MA1 till their stale entry expires. This problem is particularly severe if the gateway G exhibits this behavior. In that situation, MH1 would be temporarily unreachable even for machines outside of Network A. The easiest solution to this problem is to maintain host specific routes for MH1 (and other similar mobile hosts) with MA1 as the gateway on all machines on Network A.[3] For an overview of how gratuitous ARP processing and selective disabling of ARP replies can be implemented, interested readers are referred to [6].

There are other issues that deal mostly with efficiency and are considered outside the scope of the IETF Mobile-IP draft. Nevertheless, these issues are important and likely to be addressed in later versions of this software. One of these issues is route optimization, *i.e.* avoiding always going through the home agent when communicating with a mobile host that is away [8]. The way our software is implemented, it automatically performs one kind of routing optimization. When two mobile hosts are visiting the same foreign network, communication between them does not go through their home networks since the foreign agent has local routes for both of them. This "feature" will go away when the new IPIP decapsulator (one which sets up and tears down the local route to MH1 on a packet-by-packet basis) is used.

Mobility support at the network layer level raises interesting issues at the transport level. Currently, TCP implicitly assumes that all packet losses are due to congestion and it responds to these losses by throttling its transmission rate and recovering slowly. In a mobile environment, host migration results in a temporary loss of routes (till the registration process is successfully completed) and may cause some packets to be dropped. Since these routes heal quickly, throttling the transmission rate results in unnecessary underutilization of bandwidth. In general, what is needed is a mechanism to help the TCP layer on the sender distinguish between packet losses due to congestion (which can not be relieved immediately) and mobility (when routes can be healed more rapidly). Interested readers are referred to [1] and references therein for proposed solutions to TCP-related problems.

In the documentation accompanying version 0.95, we described the following observation when testing telnet and rlogin connections originating at the mobile host: "If any key strokes are entered when the mobile host is on the move (neither connected to its home network nor to a foreign network), the client program hangs and does not recover even after the move has been completed. On the other hand, if no key strokes are entered during the move, the session recovers flawlessly when the move is completed. Interestingly enough, we do not encounter this problem if the telnet/rlogin sesion is started from another host to the mobile node. We have not had the chance to investigated this problem in great detail but suspect that it is specific to Linux".

It turns out that this problem was indeed Linux specific. Earlier 1.3 kernels (e.g. 1.3.30) cached routing information with the first transmission of each TCP packet. Route changes that forced TCP retransmissions based on stale routing information resulted in hung connections and even kernel panics. Newer 1.3 kernels (e.g. 1.3.58) do not suffer from this problem.

It has been brought to our attention that configuring *tunnel* interfaces on home agents running *gated* may cause routing loops. Our network administrators require us to use static routing on gateways and we have not encountered this problem in that situation.

[3] For Figure 1, this can be accomplished by using the command `route add -host 128.226.3.30 gw 128.226.3.28`

7 Contacting the Authors

Several other researchers including those Rutgers University, UCLA and NEC's Computers and Communications Research Lab in Princeton have been using different versions of this software. Our implementation of Linux Mobile-IP has been successfully demonstrated at ACM's First International Conference on Mobile Computing (MobiCom'95) held at Berkeley, CA and Nomadic'96 held at San Jose, CA. As such, we believe this version to be free of serious bugs. Nevertheless, if you encounter any problems while using this software, please let us know. All comments/queries should be directed to *mobileip@anchor.cs.binghamton.edu*. Updates regarding this software will be posted periodically to the Linux Mobile-IP homepage at http://anchor.cs.binghamton.edu/~mobileip (the IP address for this machine is `128.226.3.28`).

Standard Disclaimer As with other software of this nature, all usual disclaimers apply. In particular, we make no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

Appendix A – Using Linux Mobile-IP on a Wireless Internetwork

A wireless internetwork similar to the one shown in Figure 8 has also been used to test our software. This figure is similar to the one shown in Figure 1 except that the same computer is now acting as the Gateway and Home Agent on Network A.

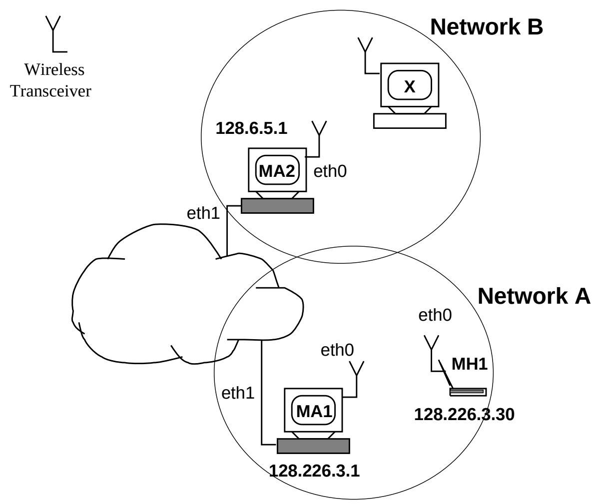


Figure 8: A wireless internetwork with mobility support.

In this case, the *agent* program on MA1 is started as

```
agent -a 128.226.3.1 -m 255.255.255.224 -h 08:1E:19:06:3B:14 -i eth0
```

where `08:1E:19:06:3B:14` is the hardware address for MA1. The *mh* program on MH1 is started as

```
mh -a 128.226.3.30 -m 255.255.225.224 -g 128.226.3.1
```

Our wireless testbed is based on NCR Wavelan cards. Both ISA (for desktops) and PCMCIA (for portable computers) versions of this card are commercially available. Bruce Janson from the University of Sydney has written a driver for the ISA Wavelan card which is included in the standard Linux kernel source code. A Linux driver for the PCMCIA card is available from Anthony Joseph at MIT. The latter requires PCMCIA card services software developed by David Hinds at Stanford. Links to these software packages can be found on the Linux Mobile-IP home page (our web site) at http://anchor.cs.binghamton.edu/~mobileip. Note that as of May, 1996 the PCMCIA Wavelan card driver works only with Linux 1.2.x. Since those kernels do not support IPIP decapsulation, mobile hosts using these cards must rely on foreign agents to decapsulate tunneled packets.

References

[1] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz. Improving TCP/IP Performance over Wireless Networks, MobiCom'95.

[2] S. Deering. Router Discovery, RFC 1256, Sep 1991. Available at ftp://ds.internic.net/rfc/rfc1256.txt.

[3] R. Droms. Dynamic Host Configuration Protocol, RFC 1541. Available at ftp://ds.internic.net/ rfc/rfc1541.txt.

[4] V. Gupta. Linux ioctl() Primer. Available at http://anchor.cs.binghamton.edu/courses/cs628/linux-ioctl.html.

[5] V. Gupta. An Introduction to the Linux 1.3.x Networking Code. Available at http://anchor.cs.binghamton.edu/courses/cs628/linux-net.html.

[6] Course projects undertaken for CS628 taught in Spring 1996 at SUNY Binghamton. Reports available at http://anchor.cs.binghamton.edu/~mobileip/Projects.

[7] Internet Engineering Task Force. Home page is http://www.ietf.cnri.reston.va.us/home.html.

[8] D. B. Johnson. Route Optimization in Mobile-IP (Internet Draft - work in progress), 1995. Latest version available from ftp://software.watson.ibm.com/pub/mobile-ip/.

[9] S. Lantinga. README.tunnel — Documentation for the Linux IPIP tunnel interface. Distributed with kernel source code in /usr/src/linux/drivers/net, Feb. 1995.

[10] Linux Documentation Project. Home page is http://sunsite.unc.edu/mdw/linux.html.

[11] C. Perkins. IP Mobility Support (Internet Draft - work in progress), 1996. Latest version available from ftp://software.watson.ibm.com/pub/mobile-ip/.

[12] R. Rivest. The MD5 Message-Digest Algorithm, RFC 1321, Apr 1992. Available at ftp://ds.internic.net/rfc/rfc1321.txt.