Solaris Mobile IP: Design and Implementation

Vipul Gupta
SUN Microsystems, Inc.
Palo Alto, CA 94303-4900

February 17, 1998

1 Introduction

In the traditional Internet architecture, a computer's IP address identifies the network to which it is attached. Datagrams are forwarded to a host[1] based on the location information encoded in its IP address. If a mobile computer moves to a new network while keeping its IP address unchanged, its address will not reflect the new point of attachment. Consequently, the underlying routing infrastructure will be unable to route datagrams to it correctly. In this situation, the mobile host must be reconfigured with a different IP address representative of its new location. Not only is this process cumbersome for ordinary users[2], but 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 (for example, ftp or telnet sessions) to be lost. Put simply, under the current Internet Protocol, if the mobile host moves without changing its address, it will lose routing; but if it does change its address, it will lose connections.

The Mobile IP working group of the Internet Engineering task Force (IETF) has developed a protocol [1, 11, 12, 13] which allows a mobile computer to be reachable at a fixed IP address (called its *home address*) irrespective of its current point of attachment to the Internet. Transport and higher layer connections are maintained across moves and all this is accomplished without the need to propagate host-specific routes throughout the Internet routing fabric. Mobile IP works with multiple data-link technologies so a computer can move between a wireless LAN and a wired LAN while maintaining its IP address and all existing connections. The protocol does not require any changes to network applications or static hosts.

The rest of this document is organized in to five sections. Section 2 is a functional overview of Mobile IP. It describes the various Mobile IP entities and their mutual interaction.

[1] In the Internet jargon, computers are often referred to as hosts

[2] Newer protocols such as DHCP [3] can simplify address reconfiguration but do not address the other problems associated with host mobility.

Readers already familiar with the protocol may decide to skip this section. Subsequent sections present details that are specific to our implementation of Mobile IP. Section 3 should be of interest to administrators and users of Solaris Mobile IP. It begins with a high level description of the main components within our binary distribution. Instructions on installing, configuring and testing the software are also included. The structure of Solaris Mobile IP source code is described in Section 4. This section is most relevant to readers that wish to make changes to our software. Section 5 lists known limitations for the current version. Finally, Section 6 includes contact information, and pointers to other helpful sources.

2 Functional Overview

The IETF Mobile IP architecture, introduces special entities called the *Home Agent* (HA) and *Foreign Agent* (FA) which cooperate to allow a *Mobile Node* (MN) to move without changing its IP address.

The term *mobility agent* is used to refer to a computer acting as either a home agent, foreign agent, or both. A network offers *mobility support* if it is equipped with a mobility agent. Mobility agents advertise their presence on a network by periodically broadcasting *Agent Advertisement* messages. A mobile node may optionally solicit an advertisement from any locally attached mobility agent through an *Agent Solicitation* message. Both these messages are based on ICMP Router Discovery [2].

Each mobile node is associated with a unique home network as indicated by its permanent IP address. Normal IP routing always delivers packets meant for the mobile node to this network. By examining the content of the agent advertisements it receives, a mobile node can determine whether it is on its home network or on a foreign network.

On detecting a move to a foreign network, a mobile node chooses a care-of address on that network and conveys it to its home agent through the exchange of *Registration Request* and *Registration Reply* messages (Figure 1). The care-of address may either belong to a foreign agent (a foreign agent care-of address), or may be acquired, perhaps temporarily, by the mobile node, e.g. through DHCP [3] or PPP [14] (a co-located care-of address).

After a successful registration, packets arriving for the mobile node on its home network are *encapsulated* by its home agent and sent to its care-of address. Encapsulation refers to the process of enclosing the original datagram inside another datagram with a new IP header. This is similar to the post office affixing a new address label over an older label when forwarding mail for a recipient who has moved. The source address in the outer header is that of the home agent and the destination is the mobile node's care-of address. This mechanism is also called *tunneling* since intermediate routers remain oblivious of the original (inner) IP header. In the absence of this encapsulation, intermediate routers will simply return packets back to the home network. The recipient of the encapsulated datagram strips off the outer header and delivers the newly exposed datagram to the mobile node. In the reverse direction, datagrams sent by the mobile node are generally delivered to their destination using standard IP routing mechanisms, not necessarily passing through the home agent. All registrations are associated with a finite lifetime. Mobility agents stop offering encapsulation

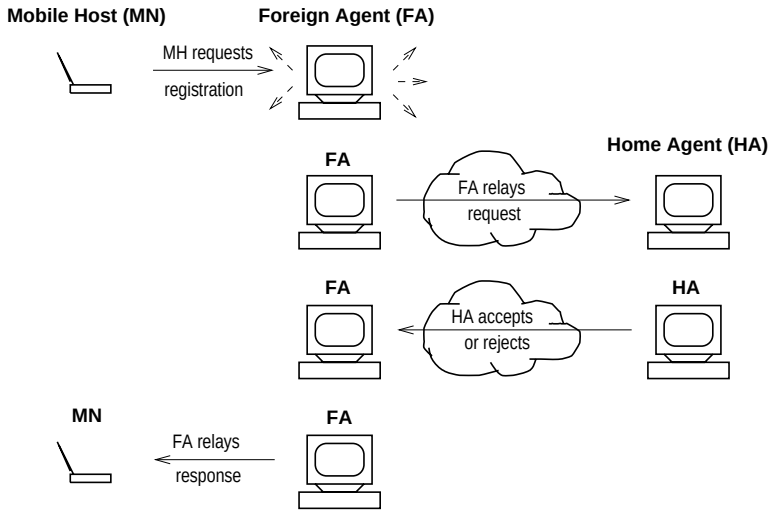

Figure 1: Mobile IP registration with a foreign agent care-of address.

and decapsulation services to a mobile node when its registration expires. A mobile node should renew its registration before it expires in order to avoid any breaks in connectivity. Registration messages are authenticated and protected from replay attacks to prevent one host from hijacking another host's packets.

Figure 2 shows the flow of datagrams to a mobile node visiting a foreign network and using a foreign agent care-of address. When a mobile node uses a co-located care-of address, it acts as its own foreign agent and performs decapsulation of tunneled datagrams. This allows a mobile node to retain its home address on a foreign network even in the absence of any foreign agents. Figure 3 depicts datagram forwarding in this case.

Upon returning to its home network, a mobile node deregisters with its home agent and operates normally without mobility services.

3 User Guide

The Solaris implementation of Mobile IP is organized as follows:

- The mobility agent software has two components – a user-level program called mipagent and a dynamically loadable kernel module called vtunl [4]. Mipagent implements both home agent and foreign agent functionality as described in RFC 2002 [11]. It responds to agent solicitations and registration messages. It also performs periodic tasks such as sending agent advertisements and aging mobility bindings and visitor entries. IPinIP encapsulation and decapsulation [12] is implemented inside the kernel by the vtunl module. Mipagent uses special ioctl commands to control the behavior of vtunl.

  Each network on which mobility support is desired should have at least one static (non-mobile) host running this software.

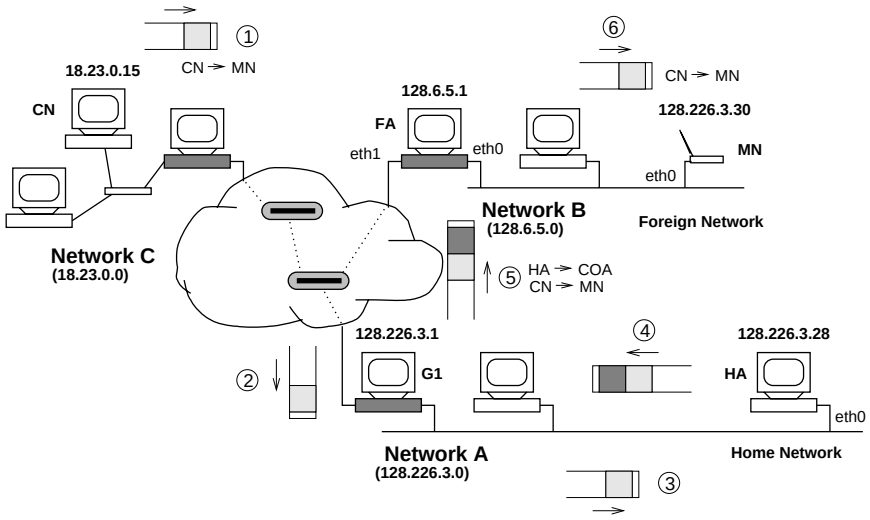

Figure 2: Datagram flow to a mobile node using a foreign agent care-of address.

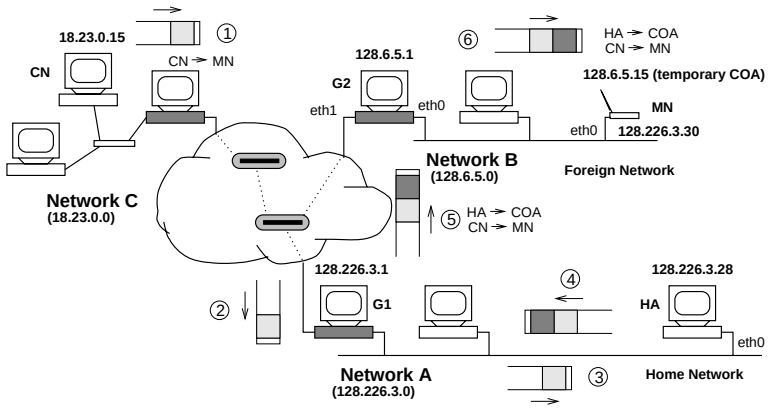

Figure 3: Datagram flow to a mobile node using a colocated care-of address.

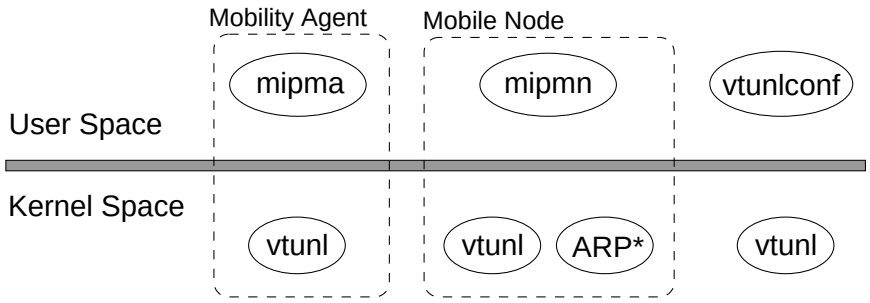

Figure 4: Major components in the Solaris Mobile IP software distribution.

- The mobile node software also has two components – a user-level program called mipmn and vtunl (described above). Mipmn implements the mobile node (aka Mobile IP client) functionality described in RFC 2002. It is responsible for sending agent solicitations, monitoring agent advertisements, detecting moves, initiating registration requests and processing replies. It also performs periodic tasks like aging registrations, retransmitting pending registration requests and sending registration renewals prior to their expiration. Vtunl is used for decapsulating IPinIP packets when the mobile node has registered a co-located care-of address. It is also used to generate IPinIP packets when a mobile node needs to set up a reverse tunnel to its home agent [8].

  This software should be run on portable computers that wish to maintain a fixed IP address irrespective of their current location.

These pieces are available from the Solaris Mobile IP home page [5] in the form of packages named `SUNWmipma` and `SUNWmipmn`, respectively. Each package also includes a program called **vtunlconf** (see Appendix C) which can be used to manipulate and/or monitor the vtunl module. It is useful for testing vtunl but is not needed for Mobile IP. Figure 4 shows the interaction between various software components.

Currently, the mobile node does not implement special ARP processing on foreign networks as described in RFC 2002. This should not impact usability of our software in most instances. Subsequent releases will include a modified ARP module as replacement for the regular ARP module bundled with Solaris. The asterisk next to the ARP module in Figure 4 indicates functionality not included in this release.

This release is an experimental prototype and using it may cause you to lose valuable information. Back up anything you consider important before installing this software. You must have either a SPARC or x86 platform with at least 16MB of RAM running Solaris 2.5.1 or later. Do not attempt to use the software on an older release of Solaris. You will need root privileges to install and start this software. As such, a certain level of familiarity with Solaris system administration is assumed. If you do not have super-user privileges on your system, contact your system administrator.

3.1 Installing the Mobility Agent Software

1. Fetch the pre-compiled package appropriate for your hardware platform.

   - SPARC: SUNWmipma-sparc.Z
   - x86: SUNWmipma-x86.Z

   Uncompress using the **uncompress** command. If you use a web browser to download these files, the browser may or may not uncompress the file for you. Some browsers may incorrectly save a compressed file without the .Z extension. In this case, you will need to rename the file with a .Z extension before using **uncompress**.

2. After you obtain the uncompressed file `SUNWmipma-sparc` or `SUNWmipma-x86`, become super-user for your machine. Use the **pkgadd** command to install this package

   pkgadd -d SUNWmipma-sparc

   or

   pkgadd -d SUNWmipma-x86

   This will create a new directory `/opt/SUNWmipma/` and subdirectories `bin`, `doc` and `drv` underneath it. The `bin` subdirectory contains binary executables (e.g. mipagent) and `doc` contains documentation. The `README` file in the `doc` sub-directory is a good place to start learning about the software. The tunneling driver vtunl (along with its configuration file `vtunl.conf`) is placed in `drv` and also copied to `/kernel/drv/`. A sample configuration file for the Mobile IP mobility agent, `mipagent.conf-sample`, is left in the `/etc/opt/SUNWmipma/` directory. You can add `/opt/SUNWmipma/bin` to your UNIX search path or make the executables and documentation available in whatever way you typically choose.

3. Copy the sample configuration file to `/etc/opt/SUNWmipma/mipagent.conf` and edit it for your local environment. See the documentation on `mipagent.conf` or comments in the sample file for details. Both are included in Appendix A. Once you have created the appropriate configuration file for your environment, execute the **mipagent** program to start the mobility agent. A UNIX-style man page for **mipagent** is also part of Appendix A.

   NOTE: The **mipagent** program manipulates kernel routing tables. You may have to kill other programs (like **in.rdisc** or **routed**) that also manipulate routing tables to prevent any adverse interaction.

3.2 Installing the Mobile Node Software

1. Fetch the pre-compiled package appropriate for your hardware platform.

   - SPARC: SUNWmipmn-sparc.Z
   - x86: SUNWmipmn-x86.Z

Uncompress using the **uncompress** command. If you use a web browser to download these files, the browser may or may not uncompress the file for you. Some browsers may incorrectly save a compressed file without the .Z extension. In this case, you will need to rename the file with a .Z extension before using **uncompress**.

2. After you obtain the uncompressed file `SUNWmipmn-sparc` or `SUNWmipmn-x86`, become super-user for your machine. Use the **pkgadd** command to install this package

   pkgadd -d SUNWmipmn-sparc

   or

   pkgadd -d SUNWmipmn-x86

   This will create a new directory `/opt/SUNWmipmn/` and subdirectories `bin`, `doc` and `drv` underneath it. The `bin` subdirectory contains binary executables (e.g. mipmn) and `doc` contains documentation. The `README` file in the `doc` sub-directory is a good place to start learning about the software. The tunneling driver vtunl (along with its configuration file `vtunl.conf`) is placed in `drv` and also copied to `/kernel/drv/`. A sample configuration file for the Mobile IP mobile node, `mipmn.conf-sample`, is left in the `/etc/opt/SUNWmipmn/` directory. You can add `/opt/SUNWmipmn/bin` to your UNIX search path or make the executables and documentation available in whatever way you typically choose.

3. Copy the sample configuration file to `/etc/opt/SUNWmipmn/mipmn.conf` and edit it for your local environment. See the documentation on `mipmn.conf` or comments in the sample file for details. Both are included in Appendix B. Once you have created the appropriate configuration file for your environment, execute the **mipmn** program to start the mobile node. A UNIX-style man page for **mipmn** is also part of Appendix B.

   NOTE: The **mipmn** program manipulates kernel routing tables. You may have to kill other programs (like `in.rdisc` or `routed`) that also manipulate routing tables to prevent any adverse interaction.

3.3 Mobile IP Testing

We recommend setting up an environment similar to that in Figure 5. It shows three networks (netA, netB, netC) with (possibly multiple) mobility agents on each and several mobile nodes (MN_i, MN_j). Having three networks allows testing of host movements between a home network and a foreign network and also between two foreign networks. One of the networks is also equipped with a DHCP server whose services can be used by visiting mobile nodes to acquire a colocated care-of address. While this example illustrates a wired network, our software works just as well in a wireless environment.

Mobile IP as defined in RFC 2002 may not operate in the presence of firewalls and/or filtering routers. Such nodes must not be present in the testbed network. Readers interested in enabling Mobile IP across firewalls, ingress filtering routers and distinct address spaces

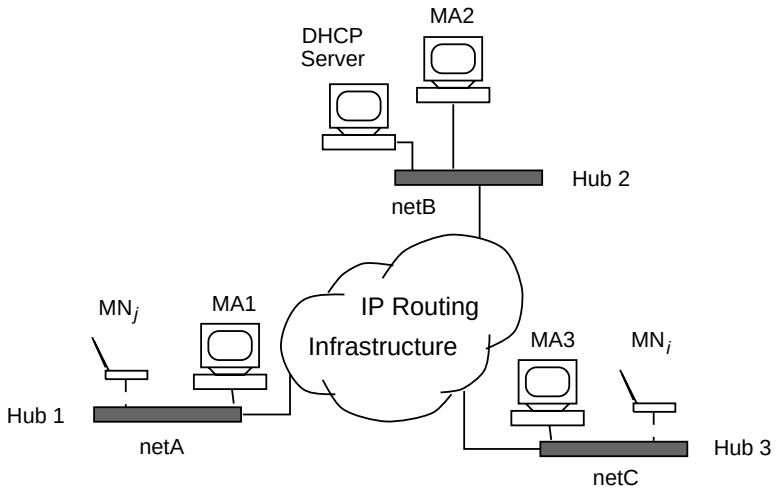

Figure 5: Sample test environment for Mobile IP.

are referred to [6, 9]. Our code implements the additional features necessary for such operation. Please refer to the manual page for `mipagent.conf` (Appendix A), and `mipmn.conf` (Appendix B) on how to activate these features.

Startup the appropriate software (e.g. **mipagent** or **mipmn**) on all mobility agents and mobile nodes. Move the mobile nodes from one network to another. In the testbed shown, this is accomplished by physically unplugging a mobile node from one hub and plugging it into another.

- Verify that the mobile node is reachable at its home address irrespective of its current location. The correspondent node may itself be a mobile node.
  - Ping the mobile node's home address continuously. The mobile node should respond at all times except when it is between moves (not connected to any network).
  - Try the same with TCP traffic (e.g. FTP/Telnet) and UDP traffic (e.g. DNS) involving the mobile node.
- Using a network monitoring tool (e.g. tcpdump or snoop) check that the mobility agents send out periodic advertisements. Make sure that these advertisements are well-formed: byte-ordering is correct for multi-byte fields, TTL is 1, at least one of H and F flags is on, B and R bits are never on unless F bit is on, sequence numbers are incremented correctly. Sequence numbers should start at zero and increment by one in each subsequent advertisement. When the value reaches 0xffff, the next value must be 256. Advertisements that have the F bit on must include at least one care-of address.
- A mobile node must send out an agent soliciatation when it starts up and also when it hasn't heard a mobility agent's advertisement for some time (the exact threshold is

user configurable). These advertisements must be sent on 224.0.0.2 or 255.255.255.255 Agents must respond to these solicitations by sending advertisements out on 224.0.0.1 or 255.255.255.255. Users can configure which of the two addresses is used for solicitation and advertisement. In our current implementation, a mobility agent responds to a solicitation by multicasting advertisements on all of its mobility supporting interfaces rather than sending a unicast message to the solicitor.

- Soon after a mobile node is moved to a new network, it must detect a move and initiate registration. A mobile node must not register with a foreign agent that has the B (busy) bit set in its agent advertisement.

- A mobile node must time out unanswered registration requests and retransmit them. The retransmission schedule is user configurable. One way to force such a retransmission (for testing) is to create a temporary network outage by unplugging the home agent.

- A mobile node registered through a foreign agent must send out a re-registration request when it detects (from the advertisement sequence number) that its foreign agent has rebooted.

- A foreign agent must respond to a registration request either by sending a denial or by forwarding it to a home agent. Denials must be sent on the local network rather than to the mobile node's home network.

- If a home agent rejects a registration request, it must send a denial indicating its reason. Denials of deregistration requests that cancel all mobility bindings must be sent on the home network, even when a tunnel has been established for that mobile node. If a registration request is accepted, the home agent must set up a tunnel for the mobile node ending at its care-of address. It must send out a gratuitous ARP mapping the mobile node's IP address to its own hardware address and start proxy-ARPing for the mobile node. It must also send out a registration reply indicating the lifetime for which the tunnel will be maintained. When a deregistration request is accepted, the home agent must undo these steps, i.e. it must stop proxy-ARPing for the mobile node, tear down the tunnel and send out a gratuitous ARP mapping the mobile node's IP address to the mobile node's hardware address.

- Mobility bindings at a home agent and visitor entries at a foreign agent must be aged periodically and deleted when the lifetime drops to zero. The foreign agent must remove pending registration requests after a certain time if no reply has been received.

- A mobile node must periodically age its active registrations and renew them before they expire. The renewal schedule is user configurable.

- On returning to its home network, a mobile node must send out a gratuitous ARP message mapping its home address to the hardware address of its own interface. It must also send out deregisteration message(s) cancelling all active bindings.

The following additional tests can be performed using a protocol testing tool such as Packet Shell [10].

- Malformed registration messages, e.g. those with multiple authentication extensions of the same type or out-of-order authentication extensions must be ignored by mobility agents and mobile nodes.
- Unrecognized extensions in the range 0-127 must be silently discarded. Unrecognized extensions in the range 128-255 should be skipped without discarding the entire message.
- A foreign agent must process and remove all MN-FA extensions in a registration request before forwarding it to a home agent. It must also remove all HA-FA extensions in a registration reply before forwarding it to a mobile node.
- A mobile node should correctly handle agent advertisements in which the ICMP router advertisement portion does not contain any addresses.
- Packets passing through the tunneling module must have their TTL decremented by 1. Packets with a zero TTL or those that appear to indicate a forwarding loop must be discarded.
- The tunneling module must participate in Path MTU discovery. If it receives a suitably large IP datagram with DF bit set, an ICMP error must be returned indicating "fragmentation required, DF set"; this message should include a suggested PMTU value that accounts for the encapsulation overhead. The tunneling module must itself monitor such ICMP messages received from other nodes and use them to maintain current estimates of MTU along different paths. The **vtunlconf** program (described in Appendix C) is capable of displaying the internal state maintained by the tunneling module.

Mobile IP without Foreign Agents

To test a mobile node's operation using a colocated care-of address, remove the mobility agent on netB. This is necessary because the current implementation of our mobile node starts using a foreign agent whenever one becomes available. With **mipmn** still running, initiate DHCP on a visiting mobile node. DHCP is included with Solaris 2.6 and the client software is invoked as follows (here we assume the mobile node uses network interface le0 to connect to netB):

```
ifconfig le0 down
ifconfig le0 auto-dhcp primary start
```

Table 1: Interoperability results from Testathon 1.

Home	Foreign Agent					
Agent	1	2	3	4	5	6
1	123567	.2....	1..5..	.....	1.35.7	1....7
2	.23...	.2....	.....	.....	.....	.....
3	1..5.7	.....	.....	7	1..5.7	.....
4	1....7	.....	.....	1....7	.....	1.....
5	1.35..	.....	...5..	.....	1235.7	.....
6	1.35.7	.....	.....	.....	1.35..	.....

Make sure that the DHCP server provides a default router as part of the DHCP exchange. When DHCP negotiation completes successfully, the mobile node will be assigned an IP address (on le0) belonging to netB and its routing table will be updated appropriately.

The mobile node program **mipmn** periodically monitors all available network interfaces. If it sees one of them configured with an IP address other than its home network, it looks in its routing table for a default router on the same network. Once it finds both pieces of information, it sends a registration request to its home agent with a colocated care-of address. No such registration is sent if a default router cannot be detected for use on the foreign network. After the registration request is accepted, **mipmn** makes additional changes to the routing table so that packets are sent out using the mobile node's home address as source.

If DHCP is unavailable, one can simulate the same behavior by using a system script that configures the mobile node for the foreign network. This includes assigning a foreign network address to one of the mobile node's interfaces and adding a default routing entry through a router on that network.

In the current implementation, a mobile node operating with a colocated care-of address may be unable to discover the hardware address of it default router on the visited network. If you encounter this problem on your network, use the workaround described in the man page for **mipmn** (Appendix B).

Interoperability tests

A previous version of Solaris Mobile IP was tested against six other Mobile IP implementations and Table 3.3 summarizes the interoperability results. Solaris Mobile IP is implementation number 3. A '3' in column 5, row 1 indicates that mobile node implementation 3 interoperated successfully with foreign agent implementation 5 and home agent implementation 1. Organization 4 did not have mobile nodes and organization 7 did not have any agents, only nodes.

Interoperability results from the Second Testahon are displayed in Table 3.3. In this

Table 2: Interoperability results from Testathon 2.

Home Agent	Foreign Agent								
	1	2	3	4	5	6	7	8	9
7					8	8	8	7,8	8
8	1,2,7,8	7,10	3		1,2,3,6,7,8,10	3,8	7,8,10		1,2,3,6,7,8,10,11
9							8,10	1,2,8,10	

table, the Solaris implementation of Mobile IP is numbered 8. Only those results in which a Solaris mobility agent or mobile node participated are shown.

The mobility agent software was also used in interoperability tests conducted at Keio University, Japan where a Solaris foreign agent supported over twenty five mobile nodes (simultaneously, at one point) running mobile node software from Stanford, Toshiba and Keio.

4 Implementation Details

Solaris Mobile IP was designed with the following goals in mind:

1. The code should be modular and simple to understand. It should have minimal dependency on standard Solaris networking code. This makes it easy to add new features to our implementation (for some suggestions, see Section 5).

2. It should be easy to port the software to another operating system. All OS-specific functionality should be cleanly separated and accessed through a well defined interface. We have successfully ported both the mobility agent and the mobile node software to Linux by replacing just one source file in each distribution. Since IPinIP encapsulation and decapsulation are already supported in the standard Linux kernel, the entire porting effort was restricted to user-level code.

Our implementation comprises two user-level programs (**mipagent** and **mipmn**), and a kernel-resident tunneling module (*vtunl*). The next two sub-sections describe the structure of the user-level code. Vtunl is described in a separate document [4].

4.1 Mobility Agent

A mobility agent uses five different data structures to keep track of its internal state:

1. A table of mobility supporting interfaces with information such as the hardware address of an interface (used for proxy-ARPing), mobility services offered on the attached network, etc.

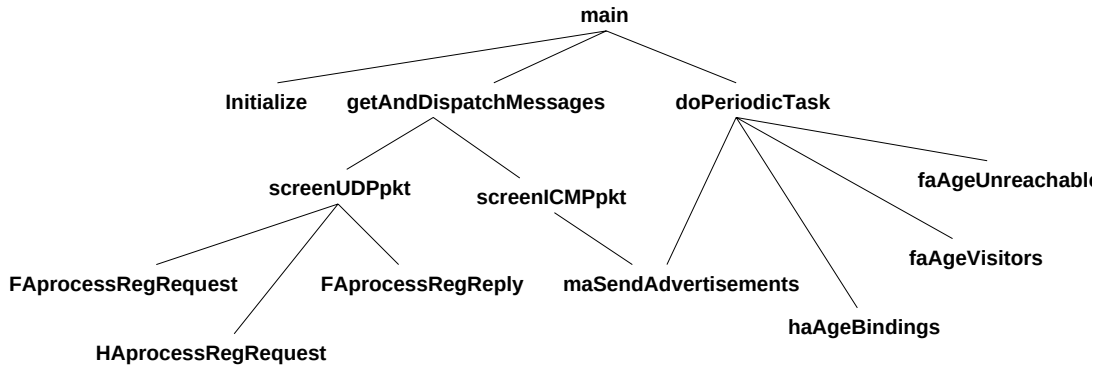

Figure 6: A high-level call graph for **mipagent**.

2. A foreign agent maintains a list of visiting mobile nodes including details of their current registration requests, e.g. source port, status (pending or accepted), lifetime (requested/granted), replay-protection ID, etc.

3. A foreign agent also maintains a list of home agents for which an ICMP unreachable message was received recently.

4. A list of mobile nodes for which it offers home agent services. Each entry contains authentication and replay-protection information for the mobile node among other things (e.g. number of active bindings).

5. A list of currently active bindings with information such as mobile node's home address, care-of address, lifetime, replay-protection ID, etc.

Figure 6 shows the high-level call graph for **mipagent**. The program reads the configuration file `/etc/opt/SUNWmipma/mipagent.conf` during initialization before entering a loop where it monitors and responds to external messages, and performs periodic house-keeping tasks.

Responding to External Messages

A mobility agent reacts to two kinds of messages: (a) ICMP messages that are either agent solicitations or carry status information about the network (e.g. a foreign agent may receive a message indicating that a home agent is unreachable), and (b) UDP messages carrying registration requests and replies. A mobility agent responds to solicitations by sending out agent advertisements. Upon receiving a registration message, a mobility agent first determines if it is a request or a reply and whether it must react to it as a foreign agent or as a home agent. A foreign agent needs to process both registration requests and replies. A home agent only needs to process registration requests.

A foreign agent that receives a registration request checks that it is properly formed, and the mobile node's service requirements can be met. If so, a new pending visitor entry is

created and the request is forwarded to the home agent. Otherwise, the foreign agent sends a denial to the visiting mobile node.

A home agent first performs a number of validity checks on any registration request it receives. If the request is poorly formed, arrives on behalf of an unsupported mobile node, or fails security checks, the home agent sends back a denial. Otherwise, an acceptance is sent. On accepting a registration, it sets up a tunnel to the mobile node's care-of address and starts intercepting and forwarding its packets. In the case of a deregistration, the home agent terminates this packet forwarding service.

When a foreign agent receives a registration reply, it tries to match it against a pending request. If no pending request is found, the reply is dropped. If the reply passes additional validity checks, it is relayed to the mobile node. Otherwise, a denial is sent from the foreign agent and the pending visitor entry is deleted. When a visiting mobile node's registration is accepted, the foreign agent reflects that new status in the mobile node's visitor entry and enables decapsulation service for the visitor.

Whenever a mobility agent rejects a registration request, it includes the reason for that denial in its registration reply.

Periodic tasks for a Mobility Agent

A mobility agent performs four different tasks periodically:

- It sends agent advertisements on local mobility supporting networks.
- It ages active mobility bindings and stops intercepting and tunneling packets for a mobile node after its binding expires.
- It ages the list of visiting mobile nodes and terminates decapsulation service for nodes whose entry has expired.
- It ages unreachability information about other home agents.

4.2 Mobile Node

A mobile node maintains the following data structures:

1. A list of active network interfaces which is periodically updated by calling OS-specific routines (on a mobile node using PCMCIA cards, network interfaces may appear and disappear dynamically). Interfaces in this list are monitored for agent advertisements and for addresses that may be usable as colocated, care-of addresses.
2. A list of active mobility agents. Upon hearing an agent advertisement, the mobile node adds its source to this table (if not already present). An agent is deleted from this list if no new advertisements have been heard recently or if the interface on which the last advertisement was heard is no longer active. Users can configure the mobile node to confirm an agent's unavailability (by soliciting an advertisement) before expiring its

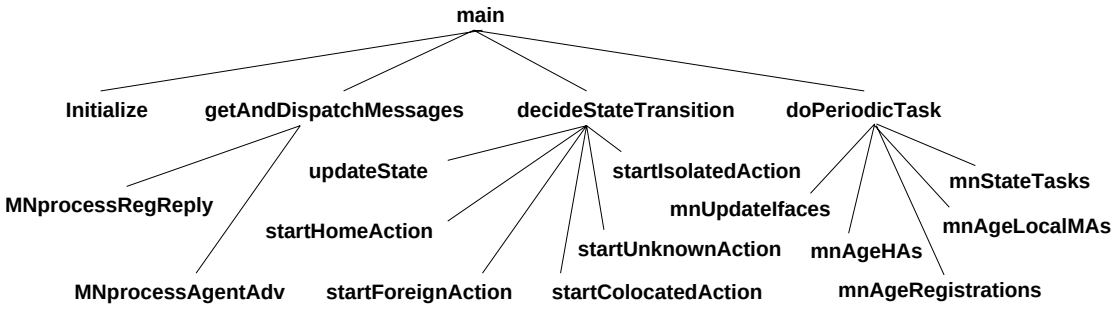

Figure 7: A high-level call graph for **mipmn**.

entry. The threshold after which an agent is considered dead is also user configurable. These thresholds should be chosen with the following trade-off in mind: smaller values allow a mobile node to react more quickly to moves at the risk of increasing network traffic (in the form of additional solicitations or registrations).

3. A list of usable home agents, including those learnt as part of dynamic home agent discovery.

4. A list of (de)registration requests including their status (pending, accepted), next time for retransmission or renewal, home agent, care-of address etc.

5. The mobile node also keeps track of the state it is currently in (see below) and state specific information.

Figure 7 shows the high-level call graph for **mipmn**. The program reads configuration information from `/etc/opt/SUNWmipma/mipagent.conf` during initialization and enters a loop. Within this loop, it monitors and responds to external messages, decides and executes state transitions, and performs periodic house-keeping tasks.

Responding to External Messages

A mobile node monitors and reacts to two kinds of messages: ICMP agent advertisements and registration replies. Agent advertisements are used to maintain the list of active mobility agents as described earlier. If an advertisement indicates that a mobile node's foreign agent has rebooted, the mobile node attempts to reregister itself.

Upon receiving a registration request, a mobile node performs some validity checks. If the reply is valid and indicates an acceptance, the mobile node reflects this status change in its list of registrations. If a mobile node receives an acceptance in COLOCATED state, it performs additional changes to its routing tables etc to complete the transition (for the other states, all required changes can be performed at the time of first transition). If the reply indicates a rejection due to a problem that can be easily corrected (e.g. replay ID mismatch), the mobile node repairs its registration request and tries again.

State Transitions

A mobile node consults its list of active mobility agents and interfaces to decide the appropriate state it should be in. There are five possible states:

HOME indicates that the mobile node is on its home network, e.g. it may have recently heard an advertisement from a home agent.

FOREIGN denotes that the mobile node has determined it is not on its home network and has detected a mobility agent offering foreign agent service.

COLOCATED means that the mobile node is not aware of any local mobility agents and has detected an address on one of its interfaces which can be used as a co-located care-of address. A mobile node should be able to pick up a default router on the visited network from its routing table; otherwise this address is deemed unusable and, hence, ignored.

UNKNOWN conveys that the mobile node has at least one active interface but none of the above states can be deduced.

ISOLATED indicates that a mobile node does not have any active network interfaces.

When a mobile node transitions to any of the first three states, it sends a new (de)registration request. This request is inserted in the mobile node's list of registrations and its network configuration is altered as follows.

On entering HOME state:

1. Flush the ARP table (except its own entry)
2. Flush the routing table (except for the loopback entry and the one set up by *vtunl*).
3. Configure the interface on which the home agent's advertisement was heard with the mobile nodes' home address and home netmask. Send a gratuitous ARP on this interface mapping the home address to its hardware address.
4. Add the default route to go through a router on the home network.

On leaving HOME state:

1. Delete the default route set through the home routers.
2. Delete the route for the home network.

On entering FOREIGN state:

1. Disable normal ARP behavior in favor of special mobile node ARP processing (not implemented).
2. Flush the ARP table (except its own entry).
3. Flush the routing table (except for the loopback entry and the one set up by *vtunl*.
4. Configure the interface on which a foreign agent's advertisement was heard with the home address and home netmask.
5. Add host specific route to the foreign agent through this interface.
6. Add a default route through the foreign agent.
7. Add an ARP entry for the foreign agent using the source hardware address picked up from its agent advertisement.

On leaving FOREIGN state:

1. Delete the default route going through the foreign agent.
2. Delete the ARP entry set for the foreign agent.
3. Delete the host specific route entry for the foreign agent
4. Restore normal ARP behavior.

On entering COLOCATED state:

This state is entered when **mipmn** detects that the mobile node is configured with an address on the foreign network. So the following changes are made after a registration for the colocated care-of address is accepted.

1. Disable normal ARP behavior in favor of special mobile node ARP processing (not implemented).
2. Create the logical clone of the interface that has the colocated care-of address. Assign the mobile node's home address and home netmask to this clone. If bringing up the clone interface creates a local route for the home network, delete that route.
3. Add a host-specific route for the default router on the foreign network (obtained in the COLOCATED state) to go through the clone interface.
4. Enable decapsulation of packets whose inner destination address is the mobile node's home address

On leaving COLOCATED state:

1. Disable decapsulation for packets whose inner source destination is the mobile node's home address
2. Delete the ARP entry for the default router and remove its route entry through the clone interface.
3. Bring down the logical clone interface.

No changes are required when entering or leaving UNKNOWN and ISOLATED states.

Periodic Tasks

A mobile node performs four kinds of house-keeping tasks:

- It periodically updates its list of active interfaces by using OS-specific services.
- It ages active mobility agents deletes them from its list if they haven't been heard from recently.
- It ages registrations potentially triggering retransmission or renewal attempts. A registration entry is deleted when its lifetime expires.
- It performs any state specific tasks, e.g. in UNKNOWN state, it sends out agent solicitations using an exponential back-off scheme.
- It ages its list of usable home agents. A home agent is deleted from this list if it was dynamically discovered and hasn't been heard from in a while (either through a registration reply or an advertisement).

5 Known Limitations

- Our software only implements IPinIP encapsulation [12], neither GRE [7] nor minimal encapsulation [13] is supported. Forwarding of broadcast packets on the home network and Van Jacobson header compression are also unsupported.
- Security associations between a home agent and a mobile node are supported but not those involving a foreign agent.
- RFC 2002 [11] requires that a mobile node visiting a foreign network must not send any broadcast ARP packets nor reply to ARP requests unless they originate from its foreign agent. This behavior is only partially implemented in the current release since full conformance requires alteration to standard Solaris code. In most situations, this omission will not have an adverse effect on Mobile IP operation.

- Mobile IP MIBs [1] have not been implemented.
- The foreign agent implementation does not support reverse tunneling [8].
- Agent advertisements sent in response to a solicitation are multicast on all mobility supporting interfaces rather than being unicast to the solicitor.
- Some routers are configured to automatically proxy ARP for any address not belonging to the local network. The presence of such routers can interfere with the operation of the Mobile IP agent. A foreign agent, as described in RFC 2002, sends out an ARP request to determine the hardware address for a visiting mobile node. If the proxy-ARPing router responds before the mobile node, forwarding loops can occur and causing the mobile node to be unreachable. Currently, the only workaround for this problem is to disable proxy-ARPing behavior at the router. A future implementation of the foreign agent will avoid ARPing for a mobile node's hardware address by picking up that information from the ethernet header of an incoming registration request.
- When a mobility agent is rebooted, it looses all Mobile IP state information, e.g. mobility bindings and visiting mobile nodes. One possible enhancement would be to save such information in non-volatile storage when **mipagent** is about to be terminated. Restoration must account for the fact that some state may no longer be active when the system comes up (e.g. the lifetime of a mobility binding may have expired).
- Newer features like mobile node operation using a PPP-assigned colocated care-of address and operation across firewall need more elaborate testing.
- The command **ifconfig -a** hangs if it tries to print hardware address information for ip0 – an interface created when the Mobile IP code plumbs vtunl between /dev/ip and the ip module. Use **ifconfig -a inet** instead to avoid printing ethernet addresses. The ethernet address of a regular interface can still be printed by specifying its name in the ifconfig command, e.g. **ifconfig le0**.

6 Contact Information

At this time, the Solaris Mobile IP software is an experimental, unsupported prototype. Feedback is welcome, however we cannot promise that we will be able to respond to your feedback.

Send feedback to *mobile-ip@lassie.Eng.Sun.COM*

Check the Solaris Mobile IP web page [5] from time to time for information regarding the status of current and future releases of this software.

References

[1] D. Cong et al., Editors, The Definitions of Managed Objects for IP Mobility Support using SMIv2, *RFC 2006*, Oct. 1996.

[2] S. Deering, Editor, ICMP Router Discovery Messages, *RFC 1256*, Sep. 1991.

[3] R. Droms, Dynamic Host Configuration Protocol, *RFC 2131*, Mar. 1997.

[4] V. Gupta, A versatile tunneling interface, distributed as part of the Solaris Mobile IP software package at *http://playground.sun.com/pub/mobile-ip/*, May 1997.

[5] V. Gupta and S. Madhani, Solaris Mobile IP, on-line home page at *http://playground.sun.com/pub/mobile-ip/*.

[6] V. Gupta and G. Montenegro, Secure and Mobile Networking, to appear in the ACM/Baltzer Journal on Special Topics in Mobile Networks and Applications (MONET).

[7] S. Hanks et al., Generic Routing Encapsulation (GRE), *RFC 1701*, Oct. 1994.

[8] G. Montenegro, Reverse tunneling for Mobile IP, Internet Draft *draft-ietf-mobileip-tunnel-reverse-05.txt* – work in progress, Jan. 1998.

[9] G. Montenegro and V. Gupta, Firewall support for Mobile IP, Internet Draft *draft-montenegro-firewall-sup-03.txt* – work in progress, Jan. 1998.

[10] S. Parker and C. Schmechel, The Packet Shell Protocol Testing Tool, available from *http://playground.sun.com/psh/*, Dec. 1997.

[11] C. Perkins, Editor, IP mobility support, *RFC 2002*, Oct. 1996.

[12] C. Perkins, IP encapsulation within IP, *RFC 2003*, Oct. 1996.

[13] C. Perkins, Minimal encapsulation within IP, *RFC 2004*, Oct. 1996.

[14] W. Simpson, The Point-to-Point Protocol (PPP), *RFC 1661*, Jul. 1994.

APPENDIX A

This appendix includes:

- UNIX-style man pages for **mipagent**,
- similar pages for its configuration file `mipagent.conf`, and
- a sample configuration file with comments.

NAME

mipagent - Mobile IP agent

SYNOPSIS

mipagent

DESCRIPTION

mipagent responds to Mobile IP (de)registration requests and router discovery solicitation messages. It implements both home agent and foreign agent functionality as described in RFC 2002. Besides responding to external messages, a mobility agent also performs periodic tasks such as aging mobility bindings and visitor entries and sending agent advertisements.

This program must be run as root. At start up, it reads configuration information from the file /etc/opt/SUNWmipma/mipagent.conf. Depending on the value assigned to the debuglevel attribute in this configuration file, mipagent prints out a continuous commentary of its actions on its standard output, e.g. with debuglevel set to 3, an asterisk character (*) is printed every time the agent advertises its presence on the network and performs other periodic tasks.

DIAGNOSTICS

The mipagent program uses the vtunl tunneling interface and exits with an error if this module is unavailable or the configuration file mipagent.conf cannot be read successfully.

NOTES

Currently security associations are supported only between a home agent and one of its mobile nodes, i.e. security associations with a foreign agent are not supported.

Some routers are configured to automatically proxy ARP for any address not belonging to the local network. The presence of such routers can interfere with the operation of the Mobile IP agent. A Mobile IP foreign agent sends out an ARP request to determine the hardware address for a visiting mobile node. If the proxy-ARPing router responds before the mobile node, forwarding loops can occur and causing the mobile node to be unreachable.

FILES

/etc/opt/SUNWmipma/mipagent.conf	Configuration file for Mobile IP mobility agent.

AUTHORS

Vipul Gupta and Sunil Madhani.

SEE ALSO

mipagent.conf(4)

C. Perkins, IP Mobility Support, RFC 2002, October 1996.

NAME

mipagent.conf - configuration file for Mobile IP mobility agent

DESCRIPTION

This is the configuration file used to initialize the Mobile IP mobility agent described in mipagent(1M). It contains seven main parts in the following order:

1) version indicator, specified by the keyword "version" followed by the version number.

2) (optional) attribute value pairs

3) number of mobility supporting interfaces

4) configuration information for each mobility supporting interface

5) number of mobile nodes to which Home Agent services are offered

6) configuration information for each such mobile node (absent if part 5 is zero)

7) (optional) information to help a home agent recognize care-of addresses that are only reachable through a firewall

Blank lines are ignored and those beginning with the hash character (#) are treated as comments. For the current version of Solaris Mobile IP software, the version number in part 1 is always one. Each attribute-value pair in part 2 is of the form:

```
keyword      value
```

The following attributes are supported (attribute names are case insensitive):

debuglevel {0|1|2|3}	Controls the verbosity of debug messages (0=severe problems, 1=unexpected behavior, 2=important events, 3=complete trace)
IDfreshnessSlack n	When using timestamps for replay protection, n seconds is the maximum skew tolerated in timestamp comparisons. Default value for n is 300.
regLifetime n	Lifetime advertised in the mobility extension of an agent advertisement in seconds. Default value for n is 300 (i.e. 5 mins)
advLifetime n	Lifetime (in seconds) advertised in the RFC1256 portion of an agent advertisement. By default n is 300.
periodicInterval n	Controls the frequency of with which periodic tasks are performed by the agent. It determines how often agent advertisements are sent and different entries aged. This interval must be less than 1/3rd of advLifeTime. The default value for n is 4 (seconds).
advertiseOnBcast {0|1}	If one, advertisements are sent on 255.255.255.255 else 224.0.0.1 is used.

In part 4, each mobility supporting interface is described on a separate line. The line lists the interface name, IP address, netmask, hardware address, advertised services flags and a boolean value (0 or 1) to indicate whether the prefix length extension is included in advertisements. The service flags are expressed as a hexadecimal byte and each bit signifies a unique feature:

0x80	Registration is required through the
0x20	Agent offers home agent services
0x10	Agent offers foreign agent services

```
    0x08    Agent offers minimal encapsulation
    0x04    Agent offers GRE encapsulation
    0x02    Agent offers Van Jacobson header
```

In part 6, each mobile node is also described on a separate line. The line lists its home address, its home agent's IP address, SPI, style of replay protection (0=none, 1=timestamps, 2=nonces), length of the secret shared with its home agent (in bytes) and the shared secret itself.

Part 7 is needed only for enabling Mobile IP across firewalls. It contains information to help a home agent recognize care-of addresses that can only be reached through a firewall. Refer to the documents mentioned under SEE ALSO.

The example section below explains the details.

EXAMPLE

The following example shows the configuration file for a mobility agent that provides mobility services on one interface (le0). On that interface it acts both as a home agent as well as a foreign agent (0x20 | 0x10) and includes prefix length in its advertisements. The mobility agent provides home agent services to two mobile nodes: 192.168.10.17 and 192.168.10.18. With the first mobile node, the agent uses an SPI of 257 (decimal), and a shared secret that is six bytes long and contains alternate bytes that are 0 and 255 (decimal). For the second mobile node, the SPI is 541 (decimal), the key is ten bytes and contains the decimal values eleven through twenty in those bytes. The first mobile node uses nonces for replay protection and the second uses timestamps.

```
# start of file
version 1
# attribute value pairs
debuglevel  3
IDfreshnessSlack 300
reglifetime  200
advlifetime  200
periodicInterval 5
# number of mobility supporting interfaces
```

```
     1
     # their description
     le0  192.168.10.11  255.255.255.0  08:00:20:11:cb:e2   30   1
     # number of mobile nodes supported as home agent
     2
     # their description
     192.168.10.17  192.168.10.11  257  2   6  00ff00ff00ff
     192.168.10.18  192.168.10.11  541  1  10  0b0c0d0e0f1011121314
     # Information about a firewall protected domain
     # number of address ranges considered ``internal''
     1
     # address ranges specified as address/netmask pairs
     192.168.10.0   255.255.255.0
     # firewall to which pkts must be directed when a care-of
     # address is ``external''. For this example, whenever a pkt
     # is destined for a care-of address outside the range
     # 192.168.10.0 through 192.168.10.255, the packet will be
     # tunneled to the firewall 192.168.10.253.
     192.168.10.253
     #end of file

FILES
     /etc/opt/SUNWmipma/mipagent.conf

SEE ALSO
     mipagent(1M)

     G. Montenegro and V. Gupta, Firewall support for Mobile IP,
     Internet Draft <draft-montenegro-firewall-sup-03.txt>, Jan
     1998.

     V. Gupta and G. Montenegro, Secure and Mobile Networking, to
     appear in ACM/Baltzer Journal on Special Topics in Mobile
     Networks and Applications (MONET).

AUTHOR
     Vipul Gupta -- vipul.gupta@eng.sun.com

SunOS 5.5.1         Last change: 12 May 1997                    x
```

mipagent.conf-sample

```
# A sample configuration file for mobility agents.
# Lines starting with the hash character (#) are treated as comments.
# Blank lines are ignored.
# It contains seven main parts (the following ordering must be preserved):
#    1. version indicator
#    2. (optional) attribute value pairs
#    3. number of mobility supporting interfaces
#    4. configuration info for each mobility supporting interface
#    5. number of mobile nodes to which HA services are offered.
#    6. configuration information for each such mobile node.
#    7. (optional) information to help a home agent recognize
#       care-of addresses that are only reachable through a firewall.
#
# Note that part 2 and 7 are optional and if item 5 is zero, item 6 need
# not be present.

# version number for the configuration file. This line is required
# and must be the first non-comment/non-blank line.

version 1

# Other (optional) attribute-value pairs. Note that the attribute names are
# case insensitive. Currently the following attributes are allowed:
#
#  debuglevel       0-3 (controls verbosity of debug messages, 0=severe
#                       problems, 1=unexpected behavior, 2=important events
#                       3=complete trace including messages)
#  IDfreshnessSLack  n  (When using timestamps for replay protection,
#                this is the maximum skew tolerated in timestamp
#                  comparisons).
#  regLifetime       n  (Lifetime advertised in the mobility extension)
#  advLifetime       n  (Lifetime advertised in the RFC1256 portion)
#  periodicInterval  n  (Controls the frequency of advertisements and
#                       the granularity of other internal timers, e.g.
#                       aging of various bindings etc). This interval
#                       must be less than 1/3 advLifeTime.
#  advertiseOnBcast  1  (If 1, advertisements are sent on 255.255.255.255
#                       rather than 224.0.0.1)
#
# The agent program initializes appropriate default values for these
```

```
# parameters in agent.c (near the start).

debuglevel  3
IDfreshnessSlack 300
reglifetime  200
advlifetime  200
periodicInterval 5
advertiseOnBcast 0

# number of mobility supporting interfaces
1

# one line for each interface containing:
#   interface name, addr, netmask, advertised services flag, and prefix flag
# The advertised services flag should be a hexadecimal number obtained
# by the logical ORing of some combination of the following.
#
#    ADV_IS_HOME_AGENT          0x20
#    ADV_IS_FOREIGN_AGENT       0x10
#    ADV_MIN_ENCAP              0x08
#    ADV_GRE_ENCAP              0x04
#    ADV_VJ_COMPRESSION         0x02
#
# It is invalid to set any service flag which is not currently implemented.
# The prefix flag is either 0 or 1 and controls whether prefix length
# extensions are included in agent advertisements (1 = include).
# In the following example, advertisements sent out on le0 offer
# both home agent and foreign agent services (0x20 | 0x10 = 0x30)
# and prefix length extensions are included.
eth0 129.146.122.191  255.255.255.0 00:60:97:8F:3B:8A     30     1
# number of supported mobile nodes
5

# one line for each supported mobile node containing:
#  addr, HA's addr on home network, SPI, Replay Prot code, key len, and key
# The replay protection code determines the replay protection
# algorithm used (0=NONE, 1=TIMESTAMPS, 2=NONCES). In the following
# example, SPI is 1, NONCES are used for replay protection and the
# key is 16 byte long with each byte being 0x11 (i.e. hexadecimal 11).
129.146.122.139  129.146.122.191  257  1  16  11111111111111111111111111111111
129.146.122.192  129.146.122.191  257  1  16  11111111111111111111111111111111
129.146.122.194  129.146.122.191  257  2  16  11111111111111111111111111111111
```

```
129.146.122.195  129.146.122.191  257  1  16  11111111111111111111111111111111
129.146.122.199  129.146.122.191  257  1  16  11111111111111111111111111111111

# If you wish to use a home agent in the presence of firewalls as
# described in ``Secure and Mobile Networking'' then you must provide
# additional information to help the home agent distinguish between
# internal and external addresses, i.e. those reachable only through
# a firewall. Currently, this configuration is very simple-minded.
# Only internal networks are specified as address/netmask pairs. If a
# care-of address lies outside all of these ranges then the home agent
# uses an IPIP tunnel to send those packets over to the specified
# firewall (currently only one firewall can be specified). In the
# following example, any care-of address which is not of the type
# 129.146.x.y is considered external and is directed at the firewall
# 192.9.207.1.

#Protected Domain Info
#No of address ranges
1

#Address Netmask
129.146.0.0 255.255.0.0

#List of firewall addresses
192.9.207.1
```

APPENDIX B

This appendix includes:

- UNIX-style man pages for **mipmn**,
- similar pages for its configuration file `mipmn.conf`, and
- a sample configuration file with comments.

TODO: Add a sample IPSec script for firewall traversal.

NAME

mipmn - Mobile IP client (aka mobile node)

SYNOPSIS

mipmn

DESCRIPTION

This program implements the mobile node functionality as described in RFC 2002. It is responsible for sending Mobile IP agent solicitations, initiating registration requests, and processing agent advertisements and registration replies. It also performs periodic tasks such as aging registrations, retransmitting pending registration requests and sending registration renewals prior to their expiration.

This program must be run as root. At start up, it reads configuration information from the file /etc/opt/SUNWmipmn/mipmn.conf. Depending on the value assigned to the debuglevel attribute in this configuration file, mipmn prints out a continuous commentary of its actions on its standard output, e.g. with debuglevel level set at 3, mipmn periodically prints a single character to identify which state it is in (H=home, F=using foreign agent, C=using colocated address, U=unknown, I=isolated from the network).

DIAGNOSTICS

The mipmn program uses the vtunl tunneling interface and exits with an error if this module is unavailable or the configuration file mipmn.conf cannot be read successfully.

NOTES

Currently security associations are supported only between a home agent and one of its mobile nodes, i.e. security associations with a foreign agent are not supported.

Since the program manipulates routing table entries, it is highly recommended that other system programs such as in.rdisc and routed which also manipulate kernel routing tables be terminated before starting mipmn.

BUGS

Some routers are configured to ignore ARP requests from hosts whose address does not belong to the network from which the request is received. The presence of such routers can interfere with the operation of this Mobile IP client. When a mobile node acquires a colocated care-of address on a visited foreign network, mipmn sends out ARP requests for the foreign router using its home address and may never get a reply. In this case, the hardware address of the foreign router must be explicitly added to the mobile host's ARP cache, e.g. this can be made an integral part of the script used to acquire the care-of address.

FILES

/etc/opt/SUNWmipmn/mipmn.conf	Configuration file for Mobile IP client.

AUTHORS

Vipul Gupta and Sunil Madhani.

SEE ALSO

mipmn.conf(4)

C. Perkins, IP Mobility Support, RFC 2002, October 1996.

NAME

mipmn.conf - configuration file for Mobile IP client

DESCRIPTION

This is the configuration file used to initialize the Mobile IP mobile node described in mipmn(1M). It contains six main parts in the following order:

1) version indicator, specified by the keyword "version" followed by the version number.

2) (optional) attribute value pairs

3) information about the home network: mobile node's home address, home netmask and default router(s) at home.

4) number of home agent entries

5) one line of information for each home agent: it includes an address, and security parameters to use when communicating with that home agent. If "default" (without quotes) is specified as one of the addresses, the corresponding security parameters are used in dynamic home agent discovery requests. Those parameters are also used when communicating with dynamically discovered home agents.

6) (optional) information to help a mobile node recognize a colocated care-of address that is outside its firewall protected domain. This part is needed only for enabling Mobile IP across firewalls. Refer to the documents mentioned under SEE ALSO.

Blank lines are ignored and those beginning with the hash character (#) are treated as comments. For the current version of Solaris Mobile IP software, the version number in part 1 is always one. Each attribute-value pair in part 2

is of the form:

```
      keyword      value
```

The following attributes are supported (attribute names are case insensitive):

debuglevel {0|1|2|3}		Controls the verbosity of debug messages (0=severe problems, 1=unexpected behavior, 2=important events, 3=complete trace)
periodicInterval	n	Controls the granularity of various internal timers, e.g. aging of registration requests.
agentSolicitThreshold	t1	If a mobile node hasn't heard from an agent in the last t1 seconds, it explicitly solicits an agent advertisement as a means of checking that agent's availability.
agentExpireThreshold	t2	If a mobile node hasn't heard from an agent in the last t2 (> t1) seconds, it considers that agent unavailable.
ifacePollThreshold	n	A mobile node checks its list of available network interfaces, once every n seconds. If an interface is detected to have an address not belonging to the mobile node's home network, the mobile node may try to use that address as a colocated care-of address. If no interfaces are found to be up, the mobile node deduces it is in isolated state.
regLifetime	n	Default registration lifetime requested in the absence of any agent advertisements, e.g. when registering a colocated care-of address. The home agent may decide to grant a smaller lifetime.

```
regRoundTrip          n   Extremely coarse round trip estimate for
                          scheduling retransmissions.

regflags              n   Discretionary flag values to be used in a
                          registration request. This should be a
                          hexadecimal number formed by ORing together
                          some combination of the following.

                              REQUEST_SIMULTANEOUS_BINDING      0x80
                              REQUEST_BROADCAST_DATAGRAMS       0x40
                              REQUEST_MINIMAL_ENCAPSULATION     0x10
                              REQUEST_GRE_ENCAPSULATION         0x08
                              REQUEST_VJ_COMPRESSION            0x40

IDfreshnessSlack      n   When using  timestamps  for  replay
                          protection, n seconds is the maximum
                          skew tolerated  in  timestamp  com-
                          parisons. Default value for n is 300.

ignoreIfaces  if1  if2    List of interfaces that should be ignored
                          by the mobile node. These interfaces are not
                          monitored for agent advertisements or
                          for a potential co-located address.

retransmissionPolicy x1 x2 ....
                          Multiples of the round trip estimate
                          regRoundTrip after which successive
                          retransmissions of a registration request
                          are sent, e.g. the first retransmission
                          occurs at x1 times regRoundTrip and if no
                          reply arrives in the intervening period,
                          the second occurs after x2 times regRoundTrip
                          and so on upto the specified maximum.

renewalPolicy  x1 x2 ...  Accepted, renewable registrations are
                          renewed when remaining time becomes x1
                          times the granted time. Another renewal
                          is sent, if necessary, when remaining
                          time drops to x2 times the originally
                          granted lifetime and so on. The values
                          x1, x2 etc must satisfy
                            1 > x1 > x2 > ... > 0
```

EXAMPLE

The following example shows the configuration file for a mobile node that has a home address 192.168.122.195, a home network mask of 255.255.255.0 and uses 192.168.122.1 as the default router at home. It is configured to use either 192.168.122.123 as its home agent or any others that are dynamically discovered.

```
# start of file
version 1
# attribute value pairs
debuglevel  3
IDfreshnessSlack 300
agentSolicitThreshold 6
agentExpireThreshold  8
regLifetime           200
retransmissionPolicy  4  8  16
renewalPolicy         0.5 0.25 0.1
periodicInterval 5
# information about the mobile node's home network: home address,
# netmask and default router(s). If the keyword "discover" (without
# quotes is specified for the default router, the mobile node will
# attempt to discover a default router. However, this functionality
# is not yet available.
192.168.122.195   255.255.255.0  192.168.122.1
# number of home agent entries, more home agents can be dynamically
# discovered and used if a default entry is specified.
2
# Information for each home agent: home agent's address, and
# security parameters including SPI (Security Parameter Index),
# replay protection mechanism (0=NONE, 1=TIMESTAMPS, 2=NONCES),
# shared secret length (key length) and the key.
# In the following example, when talking to home agent
# 192.168.122.123, the mobile node uses SPI 257, NONCE-based
# replay protection and a 16 byte long key with each byte being
# 0x11 (i.e. hexadecimal 11). In addition, the mobile node can
# dynamically discover other home agents and for them it uses
# SPI 570, TIMESTAMP based replay protection, 16-byte key with
# each byte equal to 0x22 (i.e. decimal 34). The home agents
# should be configured to use the same set of security parameters
# as expected by the mobile node.
```

```
    192.168.122.123  257  2  16    11111111111111111111111111111111
    default          570  1  16    22222222222222222222222222222222


    # (Optional) If you wish to use a mobile node in the presence of
    # firewalls as described in ``Secure and Mobile Networking'' then
    # you must provide additional information to help the mobile node
    # distinguish between internal and external addresses. Currently,
    # this configuration is very simple-minded. Only internal networks
    # are specified as address/netmask pairs. If the mobile node
    # acquires a colocated care-of address that does not belong to any
    # of the address ranges specified as ``internal'', the mobile node
    # will invoke a script called IPSecScript with three parameters:
    # the first one is either ``on'' or ``off'' (without the quotes),
    # the second is the interface name on which the colocated care-of
    # address was discovered and the third is the care-of address
    # itself.
    #
    # When called with the ``on'' parameter, it is upto the script to
    # enable whatever processing (e.g. IPSec) is necessary so that
    # packets sent by the mobile node can be authenticated at the
    # firewall and allowed to pass. These packets may also be encrypted
    # to counter eavesdropping. When called with the ``off'' parameter,
    # the script should disable such security processing.

    # number of address ranges considered part of the protected domain.
    1
    # address ranges specified as address/netmask pairs
    192.168.122.0  255.255.255.0
    #end of file

FILES
    /etc/opt/SUNWmipmn/mipmn.conf

SEE ALSO
    mipmn(1M)

    G. Montenegro and V. Gupta, Firewall support for Mobile IP,
    Internet Draft <draft-montenegro-firewall-sup-03.txt>, Jan
    1998.

    V. Gupta and G. Montenegro, Secure and Mobile Networking, to
```

```
     appear in ACM/Baltzer Journal on Special Topics in Mobile
     Networks and Applications (MONET).


AUTHOR
     Vipul Gupta -- vipul.gupta@eng.sun.com

SunOS 5.5.1         Last change: 14 Sep 1997                    x
```

mipmn.conf-sample

```
# A sample configuration file for mobile nodes.
# Lines starting with the hash character (#) are treated as comments.
# Blank lines are ignored.
# It contains six main parts (the following ordering must be preserved):
#    1. version indicator
#    2. (optional) attribute value pairs
#    3. Home Address, HomeNetmask  and default routers at home
#    4. number of home agents
#    5. configuration info for each home agent
#    6. (optional) Firewall traversal information

# version number for the configuration file. This line is required
# and must be the first non-comment/non-blank line.
version 1

# Other (optional) attribute-value pairs. Note that the attribute names are
# case insensitive and the program uses appropriate default values for
# attributes not specified in this configuration file. The unit of time is
# seconds. Currently the following attributes are allowed:
#
# debuglevel            0-3 (controls verbosity of debug messages, 0=severe
#                            problems, 1=unexpected behavior, 2=important
#                            events, 3=complete trace including messages)
# periodicInterval      n   (Controls the granularity of various internal
#                           internal timers, e.g. aging of local mobility
#                           agent table)
# agentSolicitThreshold t1  (If a mobile node hasn't heard from an agent in
#                           the last t1 seconds, it explicitly solicits an
#                           agent advertisement as a means ensuring that
#                           the agent is indeed unreachable)
# agentExpireThreshold  t2  (If a mobile node hasn't heard from an agent in
#                           the last t2 (> t1) seconds, it considers that
#                           agent unusable).
# ifacePollThreshold    n   (A mobile node checks its list of available
#                           network interfaces, once every n seconds.
# regLifetime           n   (Default registration lifetime requested in
#                           the absence of any agent advertisements).
# regRoundTrip          n   (Very rough round trip estimate for scheduling
#                           retransmissions)
# regflags              n   (Discretionary flag values to be used in a
```

```
#                          registration request)
#
#       The regflags value should be a hexadecimal number obtained
#       by the logical ORing of some combination of the following.
#               REQUEST_SIMULTANEOUS_BINDING    0x80
#               REQUEST_BROADCAST_DATAGRAMS     0x40
#               REQUEST_MINIMAL_ENCAPSULATION   0x10
#               REQUEST_GRE_ENCAPSULATION       0x08
#               REQUEST_VJ_COMPRESSION          0x40
#
#  haCooloffPeriond     n  (Time period for which to mark home agent
#                          unusable if a registration request is
#                          is rejected by home agent with a reason that
#                          can not be immediately fixed)
#  faCooloffPeriond     n  (Time period for which to mark foreign agent
#                          unusable if a registration request is
#                          is rejected by foreign agent with a reason that
#                          can not be immediately fixed)
#  IDfreshnessSLack     n  (When using timestamps for replay protection,
#                          this is the maximum skew tolerated in timestamp
#                          comparisons).
#
#  ignoreIfaces         ifname1 ifname2
#                           (List of interfaces that should be ignored
#                           by the mobile node. These interfaces are not
#                           monitored for agent advertisements or
#                           for a potential co-located address).
#
#  retransmissionPolicy x1 x2 .... (Multiples of estimated roundtrip times
#                               after which successive retransmissions of
#                               a registration request are sent, e.g. the
#                               first retransmission occurs at x1 times
#                               regRoundTrip and if no reply arrives in
#                               the intervening period, the second
#                               occurs after another x2 times regRoundTrip
#                               and so on upto a maximum.
#
#  renewalPolicy        x1 x2 ..... (Accepted, renewable registrations are
#                                 renewed when remaining time becomes x1
#                                 times the granted time. Another renewal
#                                 is sent, if necessary, when remaining
#                                 time drops to x2 times the originally
```

```
#                                granted lifetime and so on.

debuglevel              3
periodicInterval        2
agentSolicitThreshold   6
agentExpireThreshold    8
ifacePollThreshold      10
regLifetime             60
regRoundTrip            4
regFlags                0
haCooloffPeriod         30
faCooloffPeriod         30
IDfreshnessSlack        300
retransmissionPolicy    4       8       16
renewalPolicy           0.5     0.25    0.1


# Information about the mobile node's home network
# <Home Address of Mobile node> <Netmask> <default routers>
# <default routers> is either a list of IP addresses in dotted decimal or
# the keyword "discover" (without quotes).
# NOTE: Currently, only an explicit list of routers is supported. To be
#       safe, we recommend that you specify only one default router
#       instead of a list. Different operating systems deal differently
#       with multiple default routing entries.
#129.146.122.195        255.255.255.0      129.146.122.1
# MODIFY THIS FOR YOUR LOCAL ENVIRONMENT
129.146.122.195         255.255.255.0      129.146.122.1

# Number of home agent entries (entries beginning with a dotted decimal
# IP address specify a unique home agent, entries beginning with the
# keyword "default", without quotes, correspond to dynamically discovered
# home agents.
# MODIFY THIS FOR YOUR LOCAL ENVIRONMENT
2

# One line for each home agent
# HA's addr on home network, SPI, Replay Prot code, key len, and key
# HA's addr is in decimal format or "default"
# The replay protection code determines the replay protection
# algorithm used (0=NONE, 1=TIMESTAMPS, 2=NONCES). In the following
# example, when talking to home agent 129.146.122.191, the mobile node
```

```
# uses SPI 257, NONCE-based replay protection and a 16 byte long key
# with each byte being 0x11 (i.e. hexadecimal 11). In addition, the mobile
# node can dynamically discover other home agents and for them it uses
# SPI 570, TIMESTAMP based replay protection, 16-byte key with each byte
# equal to 0x22 (i.e. decimal 34).
# 129.146.122.191  257  2  16    11111111111111111111111111111111
# default          570  1  16    22222222222222222222222222222222
# Make sure that the home agents have been appropriately configured
# to support this mobile node and both use the same set of security
# parameters for mutual communication, i.e. the home agent
# 129.146.122.191 should be configured to support 129.146.122.195 as
# a mobile node and use the same SPI, replay protection method and
# shared secret as specified for 129.146.122.191 in this file.
# MODIFY THIS FOR YOUR LOCAL ENVIRONMENT
129.146.122.123    257  2   16   11111111111111111111111111111111
default            570  0   16   22222222222222222222222222222222

# (Optional) If you wish to use a mobile node in the presence of firewalls
# as described in ``Secure and Mobile Networking'' then you must provide
# additional information to help the mobile node distinguish between
# internal and external addresses. Currently, this configuration is
# very simple-minded. Only internal networks are specified as
# address/netmask pairs. If the mobile node acquires a colocated
# care-of address that does not belong to any of the address ranges
# specified as ``internal'', the mobile node will invoke a script
# called IPSecScript with three parameters: the first one is either
# ``on'' or ``off'' (without the quotes), the second is the interface
# name on which the colocated care-of address was discovered and the
# third is the care-of address itself.
#
# When called with the ``on'' parameter, it is upto the script to
# enable whatever processing (e.g. IPSec) is necessary so that packets
# sent by the mobile node can be authenticated at the firewall and
# allowed to pass. These packets may also be encrypted to counter
# eavesdropping. When called with the ``off'' parameter, the script
# should disable such security processing.

# number of address ranges considered part of the protected domain.
1
# address ranges specified as address/netmask pairs
129.0.0.0  255.0.0.0
```

APPENDIX C

```
vtunlconf(1M)          Maintenance Commands          vtunlconf(1M)

NAME
      vtunlconf - Interactively test the vtunl tunneling driver.

SYNOPSIS

      vtunlconf <IP address> [list of interface names]
      vtunlconf -n

DESCRIPTION
      vtunlconf is used to interactively monitor the internal
      state of the vtunl tunneling driver and exercise all of
      its built-in functionality. It must be run as root. The
      program has two modes as indicated in the synopsis.

      In the first mode, vtunlconf is supplied with an IP
      address belonging to one of the configured network
      interfaces of the system and a list of all network
      interfaces from which broadcast or multicast traffic
      needs to be picked up for tunneling through vtunl.
      In this mode, vtunlconf plumbs at least two instances
      of vtunl: a module instance is plumbed on top of the
      IP driver (/dev/ip) and underneath the IP module (ip),
      a driver instance is plumbed directly underneath a
      stream head (for passing ioctls to vtunl). In addition,
      for each interface in the argument list, vtunl is plumbed
      over that device and underneath the IP module.
      This mode should be used when no other program (e.g.
      agent(1M)) has vtunl plumbed between the IP driver and
      IP module.

      In the second mode, vtunlconf assumes that another
      program (e.g. agent(1M) or vtunlconf (1M) running in the mode
      above) has plumbed vtunl between the IP driver and IP module.
      In this mode, vtunlconf plumbs only one instance of vtunl
      directly underneath a stream head. This stream is used to
      pass ioctl commands to vtunl.

      vtunlconf implements the following interactive commands.
```

```
The notation [a|b|c] indicates that exactly one of a, b,
c MUST be specified. The notation {a|b|c} is similar except
a is assumed if no choice is explicitly specified.

  encapadd [<target>|default] <tunnel_exit> <flags>
          Encapsulate pkts for target to tunnel_exit.
          Flags = 80 to preserve existing bindings,
          else 0.

  encaprem [<target>|default] [<tunnel_exit>|all]
          Stop encapsulating pkts for target to
          tunnel_exit.
  encaprem all
          Stop all encapsulation services.

  decapadd <target> {dst|src|either}
          Decapsulate pkts if target appears as specified
          in the inner IP header of an encapsulated datagram.
  decapadd all
          Decapsulate all encapsulated packets.

  decaprem [<target>|all]
          Stop decapsulation service as indicated.

  debuglevel [0|1|2|3]
          Control what is reported (0=Errors,1=Unexpected
          behavior, 2=Changes in internal state,3=Full trace
          of events) on system console.

  pmtubound [<tunnel_exit>|default] <max_mtu>
          Set a maximum PMTU for specified exit. If the default
          keyword is used, the command sets the max PMTU for
          all subsequently created tunnel exits.

  dump
          Print internal configuration information for vtunl.

  help
          Prints this information.

  exit
          Terminate vtunlconf. All instances of vtunl
```

```
plumbed by this invocation of vtunlconf are
automatically unplumbed.

EXAMPLE
     If your system is connected to the network through le0 with
     IP address 192.168.3.17 and no other program has plumbed an
     instance of vtunl between the IP driver and module, invoke
     vtunlconf as

          example# vtunlconf 192.168.3.17 le0

      This will create a point-to-point interface ip0 between
     192.168.3.17 and 255.255.255.254 and another interface le1.
     These interfaces can be listed by the ifconfig -a inet
     command. A routing table entry will also be created for
     255.255.255.254 through ip0.

     If another active program has already created ip0, invoke
     vtunlconf with the -n option.

     The vtunlconf prompt appears as

          vtunlconf>

     To get a list of available commands (described above), type
     help at the prompt.

          vtunlconf> help

     To start encapsulating packets for 192.168.3.16 to 192.168.3.100
     use the encapadd command.

          vtunlconf> encapadd 192.168.3.16 192.168.3.100 0

     This will automatically insert a new routing table entry
     equivalent to that created by
          route add 192.168.3.16 255.255.255.254 1
     The purpose of this entry is to force packets for 192.168.3.16
     through vtunl. The encapadd command also changes the internal
     vtunl configuration to prepend a new IP header with destination
     192.168.3.100 on datagrams destined to 192.168.3.16.
```

To examine the internal state of vtunl, use the dump command

```
vtunlconf> dump
                E N C A P S U L A T I O N     I N F O.

     Number of encapsulation targets = 1.

     Target          Exits   Addresses of tunnel exits
     ------          -----   -------------------------
     192.168.3.16    1       192.168.3.100,

     Number of tunnel exits = 1.

     Tunnel          Curr    Max
     exit            PMTU    PMTU    State
     ----            ----    ----    -----
     192.168.3.100   1500    1500    0x   7


                D E C A P S U L A T I O N     I N F O.

     Number of decapsulation targets = 0.
```

To test if packets are indeed tunneled as expected, ping 192.168.3.16 and monitor network traffic (e.g. using snoop or tcpdump). You should see encapsulated packets (proto 4) go out to 192.168.3.100.

To get a log of all operations performed by vtunl, use the debuglevel command with a non-zero argument (higher values result in more verbose messages)

```
vtunlconf> debuglevel 3
```

Try sending a packet to 192.168.3.16 again and monitor the console. You should see output similar to

```
vtunl_wput_dlpi: got DL_UNITDATA_REQ
IPIP: <src=192.168.3.17 dst=192.168.3.16 proto=.. len=..>
tunneled to 192.168.3.100
```

By default, vtunl will not decapsulate any IPIP datagrams it

receives, e.g. if we arrange to send an IPIP packet to this system, the console will display a message similar to

```
vtunl_decap() called
Outer src=a.b.c.d dst=192.168.3.17 proto=4  len=..
Inner src=p.q.r.s dst=w.x.y.z      proto=.. len=..
Decapsulation denied for [p.q.r.s --> w.x.y.z]
```

The following example shows how to enable decapsulation of IPIP packets in which the inner src is 192.168.3.35.

```
vtunlconf> decapadd 192.168.3.35 src
```

Executing the dump command will reflect this change in internal state.

```
vtunlconf> dump
               E N C A P S U L A T I O N     I N F O.

                          .....

               D E C A P S U L A T I O N     I N F O.

      Number of decapsulation targets = 1.

      Target          Flags
      ------          -----
      192.168.3.35    src
```

To turn off vtunl-related messages on the console, use the debuglevel command with zero as its argument.

```
vtunlconf> debuglevel 0
```

To cancel tunneling of packets with destination 192.168.3.16 to 192.168.3.100, use the encaprem command (the host route entry for 192.168.3.16 will be automatically removed).

```
vtunlconf> encaprem  192.168.3.16  192.168.3.100
```

To terminate vtunlconf, use the exit command (you will be returned

```
     to the system prompt).

          vtunlconf> exit
          example#

SEE ALSO
     V. Gupta, A Versatile Tunneling Interface, May 1997.

     This document describes vtunl in great detail including how it
     is plumbed in the Solaris STREAMS framework. It contains the
     background necessary to fully understand vtunlconf's operation.

BUGS
     Include the inet option when using the "ifconfig -a" command as
     root. In its absence, ifconfig will hang trying to print the
     ethernet address for ip0.

     When vtunlconf terminates, it does not automatically restore
     the routing table to its previous state.

     vtunlconf does not manipulate the routing table entries
     correctly for certain variations of the enapadd and encaprem
     commands (those that use the default and all keywords). A
     suitable warning is printed in such cases.

NOTES
     vtunl's internal state outlives vtunlconf or other programs
     that manipulate it. This configuration can be cleaned up
     by using the "encaprem all" and "decaprem all" commands
     inside vtunlconf. Alternatively, one may remove the vtunl
     driver and add it again by executing the following:

          example# rem_drv vtunl
          example# add_drv vtunl

AUTHOR
     Vipul Gupta -- <vipul.gupta@eng.sun.com>


SunOS 5.5.1         Last change: 12 May 1997                    x
```