This paper appears in the Proceedings of the 1996 USENIX Technical Conference, San Diego, CA, January 1996.

# Supporting Mobility in MosquitoNet

Mary G. Baker, Xinhua Zhao, Stuart Cheshire, Jonathan Stone
*Stanford University*

## Abstract

The goal of the MosquitoNet project is to provide continuous Internet connectivity to mobile hosts. Mobile hosts must be able to take advantage of the best network connectivity available in any location, whether wired or wireless. We have implemented a mobile IP system that supports seamless switching between different networks and communication devices. In contrast to previous approaches to mobile IP, we believe mobile hosts should not assume any explicit mobility support from the networks they visit, aside from basic Internet connectivity. This decision places extra responsibilities on the mobile hosts themselves. In this paper, we describe the design and implications of such a system. Measurements of our implementation show that the inherent overhead to switch networks (below 10ms) is insignificant compared to the time required to bring up a new communication device.

## 1. Introduction

We envision that ubiquitous network connectivity will someday be a reality. In the future, the Internet will be a collection of different services, both wired and wireless, often with overlapping areas of coverage. Through the combination of these services, especially with the rapid growth of wireless networks, it will almost always be possible for a mobile host to remain connected to the Internet, or at least to reconnect when so desired.

Based on our vision of this future global internetwork, the MosquitoNet project has been working on supporting continuous (or seemingly continuous) connectivity. By continuous connectivity, we mean that a mobile host can send and receive packets whenever it wishes, though the available quality of network service may vary widely.

We believe continuous connectivity is not only feasible but also crucial to make the most out of portable computers. More personal computer users are using their computers mainly for communication. It is also clear that consumers want continuous connectivity available to them, as shown by the recent introduction of two-way pagers and "500 number" phone numbers that allow consumers to receive telephone calls wherever they go, without informing callers of any change in phone number.

While the physical infrastructure for ubiquitous network connectivity will be available, there are several problems mobile hosts must overcome to make full use of it. This paper addresses two of these problems. The first is that mobile hosts must be able to switch seamlessly between different types of network devices, and the second is that mobile hosts must be able to visit foreign networks that do not provide any support for mobility.

We must be able to switch seamlessly between different network devices to take advantage of whatever connectivity is available. For example, we may need to switch from an Ethernet connection to a radio modem as we leave our offices, taking our computers with us. If we arrive at a site where there is a higher speed connection, we may want to switch once again to take advantage of it, even if the wireless service is still available.

When switching between these networks, it is important to maintain all current network conversations. Restarting all applications every time we change locations is unacceptably annoying. This is especially true for applications that run for extended periods of time and build up nontrivial state, such as remote logins with active processes. As another example, we may have selected a long thread of postings from a newsgroup and wish to read them after we move to a different location. We do not want to restart the news reader and mark the thread again. If we do not support this seamless switching between networks, we would need to rewrite all these applications to save and restore their own states. In MosquitoNet, we make this seamless switching possible without requiring changes to existing applications on mobile hosts or on the hosts corresponding with them.

The second problem we address is how to maintain connectivity when visiting foreign networks that do not explicitly support mobility for visitors. A foreign network is one operated by authorities other than those operating a mobile host's home network. For the foreseeable future, the global network will continue to be a functioning anarchy, i.e., a collection of services under different authorities. For many organizations, there is little motivation to expend much effort solely on the behalf of mobile visitors. For this reason, our mobile hosts do not require any mobility support from the networks they visit. It is this issue of mobility support in foreign networks that distinguishes the emphasis of our work from previous work on providing host mobility.

Even if future networks adopt a standard mobility protocol, our system provides mobility in the current network. It took ten years for IP multicast to reach its current stage of deployment. We do not want to wait another ten years for mobile IP support that assumes the existence of "agents" operating on a mobile host's behalf in every network it visits. MosquitoNet mobile hosts do not require such changes or additions to network infrastructures outside their home domain.

To solve these problems and gain more experience with mobility, we have designed and implemented a system that requires support only in the home domain of the mobile host and on the mobile host itself. While our approach simplifies the system in some ways, it raises design issues for the mobile host's network software. Our software must now support the mobile host's interactions with foreign networks as well as its interactions with its home network. Determining where we can or should keep mobility transparent to the mobile host's networking software is more complex in our system than in systems with foreign network support.

This paper presents our protocol, our resolution of these design issues, and our system's performance. Our measurements show that switching between available networks causes little disruption to applications running on mobile hosts or on the hosts corresponding with them.

The next section describes the differences between our approach and the previous related work. Section 3 presents our mobile IP system design and implementation. Section 4 provides some performance evaluation of the system. Section 5 describes what we have learned as a result of implementing this system, including the advantages and disadvantages of operating without agents in foreign networks, and transparency issues for mobile IP implementations. Section 6 describes some future work for our project. The final sections give some concluding remarks, together with release information for our software.

## 2. Comparison with Previous Work

Several systems have been proposed (and some implemented) that provide host mobility in the Internet. Existing Internet routing protocols are used in all these systems. As illustrated in Figure 1, the systems share several components with our approach. A *mobile host* (MH) is a host that can be reached through a constant *home IP address* regardless of its current location. A correspondent host (CH) is a host that communicates with a mobile host. The correspondent host could itself be mobile. Another component in common is a stationary host, called a *home agent* (HA). The home agent takes packets from correspondent hosts addressed to a mobile host's home IP address and forwards them to the mobile host's current point-of-attachment. This point-of-attachment in a foreign network is often called the mobile host's *care-of address*. The home agent typically forwards packets to the mobile host by *tunneling* them. This

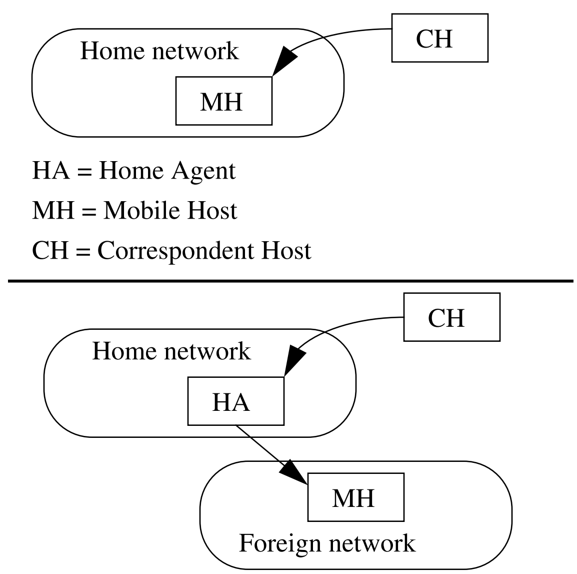


**Figure 1. Basic mobile host scenario:** The top half of the figure shows a correspondent host communicating with a mobile host that is still on its home network. The bottom half of the figure shows the path packets take when the mobile host moves to a foreign network. The correspondent host continues sending packets to the mobile host's home IP address. An agent in that network (the home agent) takes responsibility for forwarding these packets to the mobile host's new location on the foreign network.

means the home agent encapsulates each packet with an extra IP header that directs the packet to the mobile host's current care-of address. Once received, this packet must be *decapsulated* to strip off the outside header before delivery to an application on the mobile host. In this way the application receives a packet that looks like a normal packet it would receive while on its home network.

The main point that distinguishes our approach from these other systems is how much support is demanded from the foreign networks a mobile host visits. Previous work usually assumes the existence of a *foreign agent* (FA) in each network the mobile host visits. The foreign agent serves as a temporary point-of-attachment for any mobile hosts visiting its network. This means that the mobile host's home agent tunnels packets to the foreign agent, which then decapsulates them and hands the original packets directly to the mobile host on its network. The IP address of the foreign agent becomes the care-of address for the mobile host.

In contrast, our approach only requires of the host network its ability to provide a dynamically-assigned temporary IP care-of address for the mobile host itself. This IP address could be assigned by hand, but this functionality is more easily provided automatically by DHCP [4] or other link-level address negotiations such as those used by PPP and SLIP services. Without a foreign agent, networking software in the mobile host decapsulates the tunneled packets. In effect, we have collocated a simple foreign agent on the mobile host itself. The difference in the assignment of care-of addresses between our design and designs that use foreign agents is further illustrated in Figure 2. We describe the advantages and disadvantages of leaving out the foreign agent in more detail in Section 5.1.

Below we compare some previous mobile IP systems to our work.

The Columbia system [6] takes the so-called 'embedded network' approach. The approach requires a special kind of router, called a Mobile Support Router. The Mobile Support Routers work closely together to make the partitioned physical networks (called cells) appear as a single subnet. While it is optimized for localized mobility, it is hard to scale beyond the scope of a single organization such as a university.

The IMHP (Internet Mobile Host Protocol) [13] and the Harvard system [2] are roughly the same, though they were developed independently. The strong point of the proposals is that they can provide host mobility over a wide area and converge to an optimal route efficiently. But to obtain optimal routing, the MH's previous FA, its CHs, and the routers along the way (called cache agents) are used to cache the location binding for the MH. The problem with this is that it requires adding this extra support to many entities in the Internet.

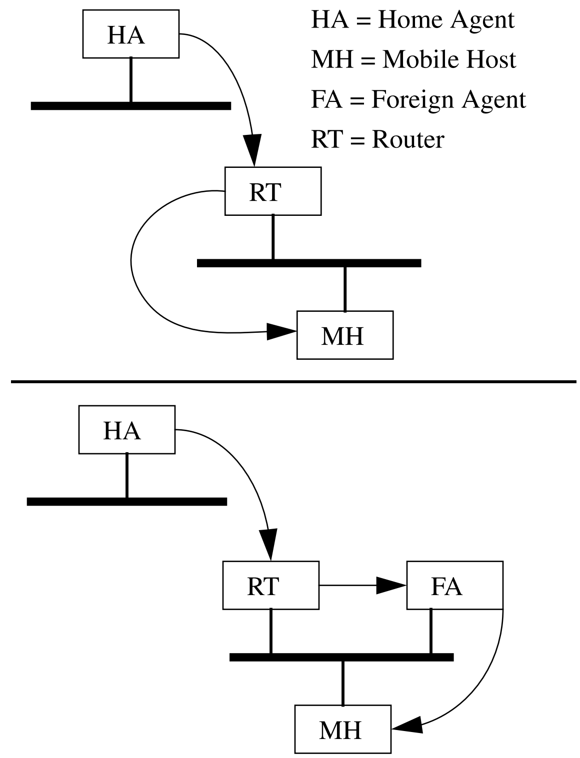


**Figure 2. Care-of addresses:** The top of the figure shows the home agent forwarding a packet to the mobile host on a network without a foreign agent. The destination address, or care-of address, for the packet is a temporary IP address on the foreign network. The router contains a map entry from this IP address to the hardware address of the mobile host's interface. The bottom of the figure shows a home agent forwarding a packet to a mobile host on a foreign network with a foreign agent. In this case, the mobile host's care-of address is the IP address of the foreign agent. The router hands the packet to the foreign agent, which then delivers it to the mobile host. In this case, the foreign agent contains a map entry that translates from the mobile host's home IP address to its hardware address.

VIP [16] places even more requirements on the existing infrastructure of the Internet. Its key point is to separate logical identifiers from physical identifiers. The network layer is divided into two sublayers: a Virtual Network Sublayer and a Physical Network Sublayer. The virtual to physical mapping information of migrating hosts is cached in Address Mapping

Tables on source hosts or intermediate routers for address resolution. Entries in these tables are updated (created or invalidated) by control packets. This technique can be applied to other identifiers, such as group identifiers for multicast communications, making this a general mechanism. But this approach also has the problem of requiring many changes to existing routers.

The current draft of the IETF Mobile IP Working Group [10] is similar to the IMHP and the Harvard system, but it differs from them in that it treats the support for optimal routing as an extension. In the proposal, the home agent has primary responsibility for processing and coordinating mobility services, while the foreign agent only has a passive and minimal role. Although a full foreign agent is expected to do more, the protocol only requires it to relay registration requests (change-of-location notifications) from the mobile host to its home agent and decapsulate packets for delivery to the mobile host. The protocol does not require any extra code on systems other than the mobile hosts, home agents and foreign agents.

We chose to base our implementation on the IETF specification, because it entails the fewest unrealistic expectations about the amount of support required from other hosts and routers in the Internet. However, we have reduced these expectations even further by leaving the foreign agent out of our basic protocol. While the most recent draft of the IETF specification suggests that the mobile host could use a dynamically acquired IP address instead of the foreign agent as its care-of address, it does not discuss the design implications of this approach, and it still encourages the use of full foreign agents.

Several of the above systems include security mechanisms. We believe that strong security is as necessary for mobile IP as it is for all networking software, but a full discussion of mobile IP security issues is beyond the scope of this paper. We do not yet implement any special security measures in our system. Although many people worry that mobile computing poses special security risks, the majority of the perceived problems are existing problems of the entire Internet that are simply brought into much sharper focus by the advent of mobile hosts.

## 3. MosquitoNet Mobile IP Design

In this section we describe the current design of our mobile IP system. We start by describing the roles of the mobile host, home agent and correspondent hosts and how they implement our basic mobile IP protocol. We then list possible optimizations to this basic protocol, including a simple one we have implemented. Finally, we describe the structure of our software. The software must correctly handle the basic protocol without precluding the use of the optimizations. We have implemented this design in the Linux operating system, version 1.2.13.

### 3.1 System Components

Of the three basic entities in our mobile IP system, the mobile host, home agent, and correspondent hosts, only the mobile host and home agent require mobility support. The mobile hosts require somewhat more support in our system than in implementations with foreign agents, since our mobile hosts must be able to encapsulate and decapsulate packets on their own. We consider this reasonable, because we have control over the software on mobile hosts.

The mobile host must be able to receive packets from correspondent hosts wherever it moves. To remain reachable, it must receive packets addressed to it at its home network. When at home, it directly receives these packets. When it leaves and connects to another network, these packets must be forwarded to it. To accomplish this, the mobile host needs to acquire a temporary care-of IP address from the new network (perhaps dynamically via DHCP). Since our approach does not assume the existence of a separate foreign agent in the new network, the mobile host serves as its own foreign agent and sends a registration message to its home agent to notify it of the new care-of address. At this point the home agent is prepared to tunnel any packets it receives from a mobile host's correspondent hosts to the mobile host's current care-of address, as previously illustrated in Figure 1.

The mobile host must also be able to send packets as well as receive them. At home, it sends packets in the normal fashion. While away from home, in our basic protocol, outgoing packets from the mobile host are also tunneled through the home agent to the correspondent hosts. With no foreign agent on the foreign network, the mobile host must encapsulate these outgoing packets itself. We can sometimes optimize the route for outgoing packets by sending them directly to the correspondent hosts, as described in Section 3.2.

The basic role of a home agent is two-fold. It must decapsulate packets sent from the mobile host for delivery to correspondent hosts, and it must encapsulate packets sent from correspondent hosts for delivery to the mobile host's care-of address. To encapsulate packets sent to the mobile host, the home agent must be able to intercept them when they arrive in the home network. To intercept these packets, the home agent must function as the ARP proxy for the mobile host upon receiving its registration request. This is done by adding an ARP entry in the home agent's own ARP cache. The home agent must then broadcast a gratuitous ARP on behalf of the mobile host to void any stale ARP cache entries on hosts in the same subnet as the mobile host's home. The home agent also adds an entry to its route table specifying that all packets for the mobile host's home IP address must be encapsulated. It adds a *mobility binding* to an internal table to record the mobile host's care-of address and other information such as the lifetime of the registration and any authentication information.

When the mobile host returns home, it de-registers with the home agent, which then removes the mobility binding and the special route table entry. The home agent should also stop functioning as the ARP proxy for the mobile host.

## 3.2 Routing Optimizations

Our basic mobile IP protocol uses a simple model of mobile networking in which outgoing and incoming packets are delivered indirectly via the home agent, using an encapsulating tunnel, to make the mobile host appear as if it were still on its home network. While this basic protocol is simple and always works, the extra path through the home agent adds latency to packet delivery. This section lists some desirable routing optimizations for outgoing packets from the mobile host, showing how they can be performed in a system without foreign agents. (We do not consider routing optimizations for the reverse path – from correspondents to the mobile host – as they are necessarily more difficult and we have not yet implemented any of them. These optimizations require the correspondent host to be able to locate the mobile host at its care-of address.)

Optimizations for packets originating from the mobile host can be evaluated based on at least three criteria: First, does the optimization improve the route a packet takes, or does it eliminate the overhead of encapsulation? (Encapsulation adds 20 bytes or more to the packet length and requires extra processing.) Second, does the optimization require some understanding of mobility on the correspondent hosts? Some correspondent hosts may be mobile themselves or may run mobile-aware software. We call these *smart correspondent hosts*, and we'd like to take advantage of them when possible. A third criterion is whether routers or firewalls are likely to object to the way a packet is addressed or sent.

To improve both the path packets take as well as eliminate encapsulation overhead, a mobile host can send packets directly to its correspondent host. This forms a *triangle route*, as illustrated in Figure 3. For this simple optimization, we set the source IP address of the packets from the mobile host to the mobile host's home IP address rather than its current care-of address. In this way, the mobile host can move as many times as it desires without the correspondent host noticing. As far as the correspondent host knows, the mobile host is always at its home address. If the source address were allowed to reflect the current care-of address, then packets with a changed address would not be recognized as coming from the mobile host without modifications to the correspondent host's software.

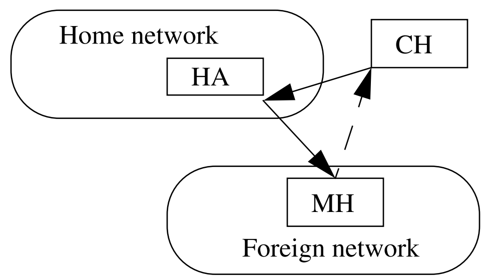


**Figure 3. Triangle route optimization:** This figure shows a simple route optimization called a triangle route. The optimization allows the mobile host to send a packet to the correspondent host without tunneling it through the home agent. This path is shown with a dotted line. The path for packets from the correspondent host to the mobile host remains unoptimized and passes through the home agent.

The problem with this optimization is that it does not work with some security-conscious routers that forbid transit traffic. Transit traffic is traffic with a source address not local to the network, as is the case with a packet using the mobile host's home IP address as source address. If the foreign network has been set up to forbid transit traffic, then the routers will drop outgoing packets from a mobile host using the triangle route optimization. This problem does not occur with the unoptimized route, which is why we have implemented both options. If we find that we cannot use the optimization, through failed attempts to "ping" a correspondent host, then we can revert to using the unoptimized route. We can cache this information for further use in the Mobile Policy Table described in Section 3.3.

A variant of the triangle route optimization, suitable for use on networks that forbid transit traffic, still sends the packet directly to the correspondent host but encapsulates the packet using the mobile host's local source IP address. Now the packet will not be dropped by the filters in the local router, because it has a valid local source address. This optimization eliminates the sub-optimal routing for outgoing packets but not the encapsulation overhead. It is appropriate when the mobile host knows that the destination host has transparent IP-in-IP decapsulation capability such as is found in recent Linux development kernels.

Finally, there are times when a mobile host wishes to talk directly with other hosts, without any attempts to hide its current physical location. This is the case when answering foreign-network management probes (such as ICMP ping and SNMP). This could also occur if the mobile host contacts correspondents for short periods knowing it will not accumulate any connection state with them. For example, the mobile host may request a web page directly from a web server. The web server simply responds and does not need to track the mobile host further. If the mobile host moves before receiving the response, the user can retry the operation. For this optimization the mobile host sends packets directly to the correspondent host, without encapsulation and without setting the source IP address to its home IP address. This is the most efficient mechanism, but it provides no mobility support. Unless the correspondent host has extra mobility support itself, it will not be able to continue communicating with the mobile host if it moves again.

Given these optimizations, a mobile host must make three decisions about how to send a packet: 1) whether to send a packet directly or tunnel it through the home agent, 2) if sending the packet directly, whether to encapsulate it, and 3) whether to use its home IP address or local address as the source address of the packet. While we only implement the triangle route optimization, it is important that our software structure not preclude these other optimizations, as described in the following section.

### 3.3 Software on the Mobile Host

Practicality dictates that we write our mobile IP code in such a way as to minimize the impact on the rest of the existing Linux kernel code. In fact, our only changes to the kernel network software are to add mobility support to IP by 1) altering the route lookup function `ip_rt_route()`, 2) adding a Mobile Policy Table, and 3) adding a virtual link-level interface, called VIF, to encapsulate packets. In this way we are able to implement both our basic protocol, allow for the previously described optimizations, and function without foreign agents.

Figure 4 illustrates the organization of our software when sending out a packet from the mobile host. The transport level protocols deliver a packet to IP, which we have extended for mobility support; our modified `ip_rt_route()` uses its Mobile Policy table combined with the usual routing table lookup to determine how the packet should be treated. The source and destination addresses of the packet are parameters to these table and function lookups. As a result, the packet may be treated as one of two basic types:

- Outside the scope of mobile IP - Some applications and services set the source address of a packet to a specific outgoing network interface, and we do not interfere with their intentions. For instance, an application may use the local-loopback interface, and there is no reason to send such packets through the home agent. Other mobile-aware applications will have their own reasons for specifying particular network interfaces. If the source address of the packet is already set, IP sends it directly to the appropriate interface, as would be done without mobile IP.
- Requiring mobile IP - If the source address of the packet is unspecified, then we must assume that the application that generated it is not mobile-aware. In this case we set its source address to the home IP address. If the application has already set the source address to the home IP address, this too means the packet is subject to mobile IP.

If we decide the packet should use mobile IP, our next decision is whether we should use any optimizations. The decision is based upon the destination address and information stored in the Mobile Policy Table. If we determine a correspondent host has extra mobility support or that a route optimization is appropriate, we cache that information in the table. In our current implementation, though, the only choices are whether to use the triangle route or to tunnel the packet through the home agent. Also, we do not yet update the table dynamically. Using the triangle route, we send the packet directly out the appropriate interface. If the packet requires encapsulation, we send it to our new virtual interface, VIF, for encapsulation.

Upon receiving a packet, VIF adds the extra IP header and sets the appropriate source and destination address in the outer (encapsulating) header. It then hands the packet back to IP for delivery to the appro-

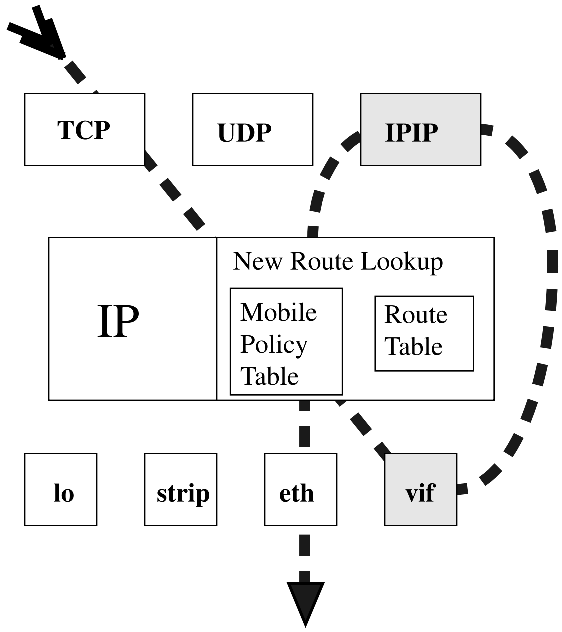


**Figure 4. Outgoing packet on a mobile host:** This figure shows the link, network, and transport layers of our protocols. The bottom (link) layer shows the device interfaces, with vif being a virtual interface that accepts packets requiring IP-within-IP encapsulation. This layer also includes the loopback interface (lo), the ethernet interface (eth), and our wireless radio interface (strip). The network layer uses the IP protocol. Besides TCP and UDP, we have added the IP-within-IP processing module (IPIP) to the transport layer. The shaded boxes indicate that vif and IP-within-IP are actually implemented as one module for efficiency. The wide dashed line shows the path an outgoing TCP packet might take in the basic mobile IP protocol.

priate physical interface. IP again looks at the packet addresses and makes its decisions, as above. To ensure the packet doesn't get encapsulated again, VIF must set the source address in the outer (encapsulating) header to a specific physical interface. In this sense, we can consider IP-within-IP (IPIP) to have delivered a new packet to IP, which treats the packet based on the same set of rules as before.

To keep the implementation simple, we have separated out routing decisions and mobility decisions. This allows us to leave the routing tables unchanged and merely add our Mobile Policy Table for IP's use. In this way, we are able to make these policy decisions by altering only a single kernel routine: the kernel's IP route lookup function, `ip_rt_route()`. This function returns, for any given destination address, both the recommended interface to use to reach that destination and the recommended source address to use. By overriding this routine, we are able to give appropriate responses to IP, TCP, and any other current or future software that may call `ip_rt_route()`. If a policy decision indicates the packet should have the home IP address as its source address, we merely return this address from the function call. If a policy decision indicates that we should encapsulate the packet, we return the encapsulating interface as the recommended interface to use. Linux's existing `ip_rt_route()` routine uses the kernel's routing tables to determine its answer; our enhanced routine additionally consults the Mobile Policy table and uses information from both tables to determine its response.

So far we have described the mechanism for sending a packet, but the mobile host must also process received packets. This is simpler than sending packets; because there are no policy or routing decisions to make, all necessary information for processing the packet is contained in its headers. The packet arrives at a physical interface and is delivered to IP. If the packet is an IP-within-IP packet, it will be decapsulated and will take the reverse of the dotted path shown in Figure 4.

## 3.4 Software on the Home Agent

The home agent shares with the mobile host the need for a virtual interface for encapsulation. This is because the home agent must encapsulate packets destined for the mobile host and tunnel them to the mobile host's current care-of address. For each mobile host away from home that has registered its current location with the home agent, there is an entry in a mobility binding list on the home agent that keeps track of information about the mobile host, such as its care-of address and the lifetime of its current registration. The home agent also adds an entry in the routing table to indicate that all subsequent packets for the mobile host's home IP address should be sent through the VIF. When the packets are sent to the VIF, they get encapsulated within another IP packet and then tunneled to the mobile host's current care-of address.

Another function of the home agent is to receive encapsulated packets from the mobile host and forward them to the correspondent host. This only involves decapsulation and IP forwarding. The decapsulation software is the same as that in the mobile host, and we simply turn on IP forwarding in the Linux kernel. In fact, more recent development versions of the Linux kernel (1.3 and later) include a decapsulation module. In these versions, we will not need to include our own.

## 4. Performance

This section describes our test-bed and contains performance data on the cost of a hand-off when a mobile host switches between different networks. Our test-bed contains both wired and wireless networks so that we can test switching between different types of device interfaces. The goal is that our wireless devices will provide a constantly available network connection, while the wired devices will provide a faster connection where available.

Our mobile hosts are Gateway 2000 Handbook486s running Linux [8]. The Handbooks are 40 Mhz subnotebooks that weigh less than three pounds and are thus very comfortable to carry around. Each Handbook has a PCMCIA card slot that we use for Ethernet connectivity with a Linksys Ethernet card. Each Handbook also has a 115.2 Kbit/second serial port that we use to connect to our wireless devices.

Our wireless devices are Metricom radio modems [14]. While these devices are commonly used to emulate a Hayes modem for point-to-point connections, Metricom also provides a connectionless datagram mode, called "Starmode," that enables a radio to send packets to any number of other radios individually. We have written a Linux driver (STRIP) that allows us to run IP over the radios' Starmode. In theory, Metricom radios can send 100 Kbits/second through the air, but in practice 30-40 Kbits/second is the best we achieve [3].

We use a Pentium 90 as the router to the home network of our mobile hosts. It is also usually used as the home agent for the mobile hosts. However, our implementation does not require the home agent to be collocated with the router; rather, we only require the home agent to be one of the hosts on the same network.

Figure 5 shows the setup of our test environment. Net 36.135 is a wired (Ethernet) subnet for our research group, serving as the home network for the mobile hosts. Net 36.8 is a wired (Ethernet) subnet belonging to the Computer Science Department and connected to the Internet. Net 36.134 is a wireless subnet for our Metricom radio devices. The results we report below are for a correspondent host located on net 36.8, but we received similar results for a correspondent host located on a campus network outside the department.

We performed two types of experiments. The first measures the disruption to the system when switching the care-of IP address of the mobile host to another IP

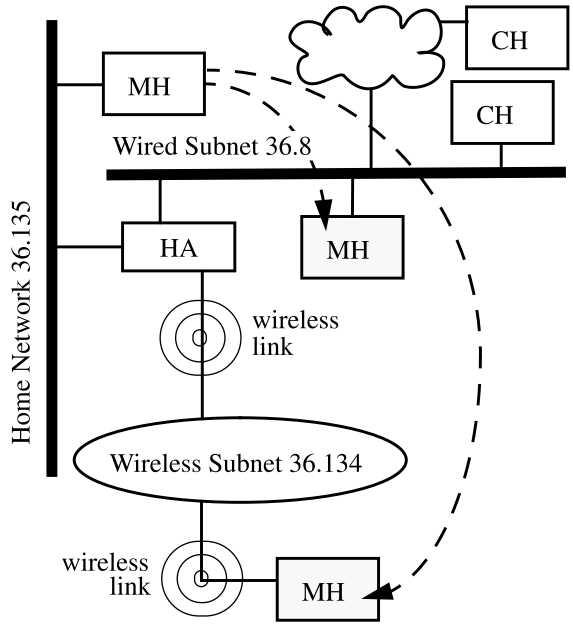


**Figure 5. MosquitoNet test-bed:** The home network for the mobile host (MH) is net 36.135. As shown by the dashed arrows, the mobile host visits either network 36.8, which it accesses using its Ethernet interface, or net 36.134, which it accesses using its wireless interface. The cloud is our artistically-challenged method for denoting the rest of the Internet. The correspondent host (CH) in this example is on net 36.8 or elsewhere in the Internet. In this figure the home agent (HA) is collocated on the router connecting our networks, but it could instead be on some other host in the home network.

address on the same wired subnet. We measure this disruption in terms of lost packets due to software overhead and address registration. Switching between addresses on the same subnet is not something we usually do in practice, but it is a measurement of the minimal essential software overhead of our system. For these tests, a correspondent host continuously sends a UDP packet to the mobile host every 10 milliseconds, and the mobile host echoes the packet back. We then measure the number of packets that were lost during the interval in which the mobile host switches addresses. This interval occurs between the time the mobile host can no longer accept packets at the old care-of address and the time it registers the new care-of address with its home agent. Packets in flight during this time will be lost by arriving at the old address. No matter how small this interval is, it is always possible for some packet in flight to arrive during this time; however, the larger the interval, the more packets we lose.

Out of the twenty iterations of this experiment, sixteen tests showed no packet loss, and the other four tests lost one packet each. This indicates that the interval during which packets can be lost is under 10 ms. We could not run the tests with a smaller interval due to the accuracy available from our current measurement tools.

The second experiment measures the disruption when switching between two types of devices, both from wired to wireless and from wireless to wired. We further subdivide this latter experiment to distinguish between *cold switching* and *hot switching*. In cold switching, we shut down one interface before starting up the other. The mobile host deletes the route to the first interface, brings the interface down, brings the new interface up, adds its route, and finally registers the new IP address with its home agent. Bringing an interface up or down usually just involves configuration in software, but some devices may also require hardware interaction. In hot switching, both of the interfaces are available and we just switch from one to the other. The mobile host merely changes its route and registers the new address with its home agent. For these tests the correspondent host sends a UDP packet every 250 milliseconds. We chose the 250ms interval because the round-trip time between the home agent and the mobile host through the radio interface is 200~250ms. Figure 6 shows our results for this second set of experiments, after running each experiment 10 times.

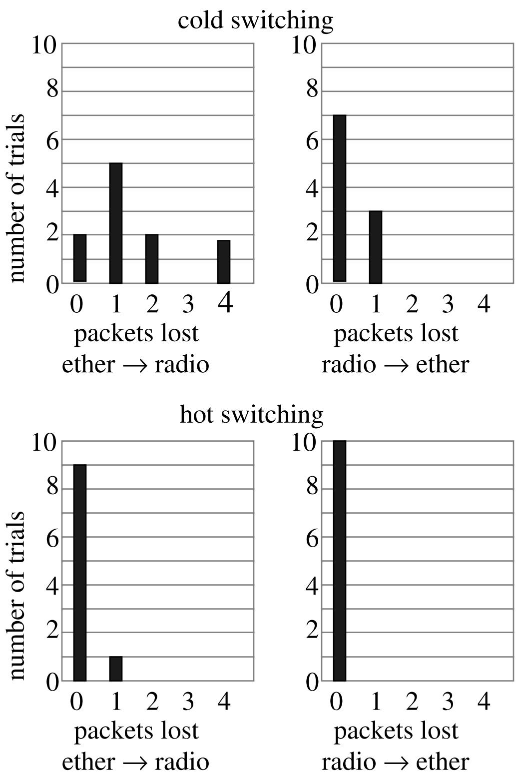


**Figure 6. Device switching overhead:** This figure shows the results from our experiments measuring packet loss when the mobile host switches between different network interfaces. We repeated each experiment ten times. The height of the bars shows the number of iterations in which the given number of packets were lost.

When doing cold switching between different devices, the period of time during which packets are lost is generally less than 1.25 seconds. The longer time interval is due to bringing up the new interface. This seems acceptable when compared with the time spent physically switching between two devices. When doing hot switching, we usually see no packet loss. (The only lost packet we observed was dropped by the radio itself and was not a result of the device switch.) This is what we would expect, since no packets should be lost if both interfaces are available.

The results for hot switching show that being able to bring up one interface before turning off the other is advantageous. Whether this is possible in practice depends on whether the user or the network monitoring software on the mobile host have any warning that connectivity is about to change. With sufficient warning, for instance, the user or the mobile host can bring up a newly available wireless interface before the old interface is disabled.

We have also collected data to break down the time in each step of the mobile host's switch to a new address and its registration with the home agent, as illustrated in Figure 7. The measurement is performed with the mobile host registering a new IP address on the same Ethernet subnet. The data reflects the average of 10 tests. The total time from start to end, including the time used to configure the interface and change the routing table, is 7.39ms. The latency between the

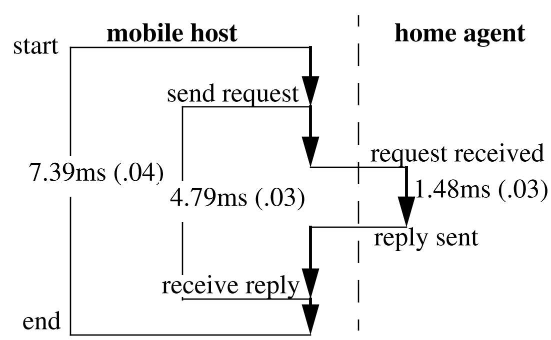


**Figure 7. Registration time-line:** The graph shows the time the mobile host spends on each step of the registration process. The total time from start to end of the address switch includes the time used in the pre-registration process (configuring the interface and changing the route table) and the post-registration process. The data reflects the average of 10 tests with standard deviations in parentheses.

mobile host sending out a registration request and receiving the reply is 4.79ms. The average time between the home agent receiving the registration request and sending out its reply is 1.48ms. The data shows that the software overhead in the registration process is small, and the home agent should be able to deal with a large number of mobile hosts simultaneously.

## 5. Designing for Mobility

Over the course of this project, we've dealt with some issues in the design of a mobile network that we believe are likely to arise in other implementations of mobility, especially those that do not use foreign agents. In this section we examine two of these issues. We first list the general ramifications of leaving out foreign agents. Next, we examine in more detail the degree to which we can hide mobility from network software running on the mobile host in the absence of foreign agents. While complete transparency sounds ideal, it is not entirely practical.

### 5.1 Leaving out the Foreign Agent

Our decision to leave out foreign agents in our basic mobile IP protocol has both advantages and disadvantages. Most of the advantages are related to the resulting reduced dependence on the foreign network. In return, though, we must acquire a temporary IP address in the foreign network, since we cannot use a foreign agent's address as our care-of address. The disadvantages of our decision are related to our use of this temporary IP address.

Advantages:

+ Main advantage: Our mobile hosts can visit networks that do not have a foreign agent.
+ Fault tolerance: The foreign agent is no longer a single point of failure for our mobile hosts' ability to continue communicating with the outside world. While we may be able to fix or restart failed home agents, we probably do not have such control over resources in a foreign network.
+ Scaling issues: We do not need a foreign agent running in every network. We only need home agents running in those networks with mobile clients.
+ Simpler protocol: Leaving out a full implementation of a foreign agent simplifies the essential part of the protocol. However, we must implement a subset of the foreign agent on the mobile hosts for decapsulating packets and registering with the home agent.
+ Compatibility with IPv6: Although mobility specifications for IPv6 have not been finalized, it appears our approach is likely to be compatible with this protocol. There will be a large IP address space, resulting in less resistance to handing out temporary addresses. Also, it appears there may be support for bindings in routers between static addresses and care-of addresses, further reducing the role of a foreign agent [12].

Disadvantages:

— Main disadvantage: The mobile host needs to acquire a temporary IP address in the foreign network. There may be networks that refuse to do this. (These same networks may also refuse to expend any other resources on visitors, including foreign agents.)

— Security: If packets for a mobile host arrive at a foreign network the mobile host has just left, those packets might be erroneously delivered to a newly arrived host that has been assigned the same temporary address the recently departed host used. This kind of accidental eavesdropping should not happen in practice because a well-written DHCP server would avoid reassigning the same IP address for as long as possible.

  Anyone concerned about deliberate malicious eavesdropping should be using end-to-end encryption rather than worrying about address reassignment problems. Packets on an Ethernet or elsewhere on the public Internet can already be easily read by a packet-sniffing program. The only security problem that is truly unique to mobile hosts is the registration of the temporary care-of address with the home agent and with smart correspondent hosts. These registrations should be authenticated with S-key, Kerberos, PGP, or some other similar strong authentication mechanism to protect against denial-of-service attacks in the form of malicious fraudulent registrations.

— Packet loss: Foreign agents may somewhat reduce packet loss. When a mobile host leaves a network, it must inform its home agent of its new care-of address. However, any packets already sent by the home agent before it receives the new registration will arrive at the old network and will be lost. If, however, a foreign agent in the old network receives the new registration before the packets arrive, it can forward the packets to the mobile host's new care-of address.

The important question, though, is whether the benefit is worth the cost. An important lesson of the Internet is that while it is relatively easy to deliver almost all packets, attempting absolute reliability makes the cost of the system grow towards infinity. Many of the components that make up the Internet exercise the option to drop packets occasionally when the cost of doing otherwise would be unreasonable. Internet protocols do not naively assume perfect delivery, but instead use end-to-end mechanisms to achieve a level of reliability appropriate to their particular needs [15]. Unless further experience with our system dictates that our potentially higher packet loss is a severe handicap, we will stick to our simple implementation.

— More complex mobile host: The lack of foreign agents somewhat complicates the design of the mobile host. This is because our mobile hosts effectively contain a simplified foreign agent. The result is that some networking software on a mobile host must be aware of its actual physical network connectivity and must handle routing operations and changes to the physical network interfaces. This gives us increased flexibility for routing optimizations but presents a new problem for our design: we must understand where mobility can be transparent to the software and where we should expose the physical network. However, this minor increase in complexity is an engineering challenge that is solvable, whereas convincing every independent authority on the Internet to provide foreign agent services would be a political task without end.

Other issues we've found that appear to be distinctions between a foreign agent implementation and our protocol do not seem significant. For instance, route optimizations handled by foreign agents are also possible in our scheme when handled by mobile hosts and home agents, as shown previously.

Despite our desire to make our basic protocol simple and self-contained, there is nothing that prevents us from implementing or using foreign agents. If it becomes appropriate, we can provide extensions to our system to implement foreign agents for visiting mobile hosts that require them. Likewise, we can extend our protocol on mobile hosts so they can take advantage of any foreign agents that happen to exist in networks they visit.

## 5.2 Design Alternatives

One of the most difficult issues we've tackled in our implementation of mobile IP is how transparent mobility should be to users and software on the mobile host itself. What we've learned through iterative designs is that complete transparency is not flexible enough and is not entirely practical. Our solution is to allow the routing tables to expose the mobile host's physical connectivity to its current network to any software or services that require this. Otherwise, we ensure that all applications on the mobile host and correspondent hosts need not know anything about mobility.

If mobility were entirely transparent to the mobile host, all of its software would have the view that it is always attached to its home network. This sounds ideal but turns out to have unpleasant implications. To hide fully whether the mobile host is at home or abroad, its routing table should always show only one unchanging interface bound to its home IP address. Even while at home this interface could not be associated with any physical network, such as the Ethernet device, since away from home the mobile host might switch to some other device for communication. The home IP address would thus be permanently bound to a virtual interface. The virtual interface would perform encapsulation while away from home and would be similar to our VIF.

The main problem with this approach is that the actual understanding of the physical interfaces would need to be hidden within VIF. This has at least three implications. First, VIF would need to construct its own real routing table showing the state of these physical interfaces. Any routing optimizations would need to be handled entirely within VIF, and this requires exposing information in the socket data structure within the interface since this information is ordinarily accessed before the packet is passed to an interface. Second, applications would not be able to bind source addresses to particular physical interfaces, because they would not be able to see the interfaces. For this same reason, applications would not be able to use two different network services at once, even if they wished to take advantage of their different characteristics for different purposes. Third, we would need to handle ICMP routing redirects differently. If a mobile host communicates with a correspondent host on the network it is visiting, the mobile host may receive routing redirects for the correspondent host that would ordinarily override any default route.

Our experience indicates that an implementation that gives up some transparency is appropriate. This is because a mobile host visiting a foreign network really has two distinct roles to play. The first is its *home* role, in which it appears virtually connected to its home network and sends packets using mobile IP features to provide transparency. Packets sent by a MH in its home role should be sent with a source address of the MH's permanent home address, perhaps with encapsulation through the virtual interface.

The second role is the mobile host's *local* role, in which it participates as a host connected to the foreign network. Examples of this second role include answering foreign-network management probes (such as ICMP ping and SNMP), and the lease-refresh of the DHCP-assigned temporary care-of address. The mobile host might also join multicast groups via the foreign network, rather than via the home network. Or it might correspond directly with another host entirely ignoring mobile IP. This is especially useful if the home agent is not reachable or has crashed.

Our belief is that a mobile IP implementation must support both roles. Foreign networks are unlikely to let visiting mobile hosts connect if the mobile hosts do not respond to local network management tools. Applications that wish to take advantage of particular network interfaces, or that use more than one interface at a time, are necessarily mobile-aware. In this sense, the packets they send are a part of a mobile host's local role. Our solution provides flexibility by exposing the mobile host's local role through the routing tables while ensuring that all applications on the mobile host and correspondent hosts need not know about the local role.

Part of the need for the mobile host to play its local role is a result of our decision to leave foreign agents out of our basic protocol. When a mobile host uses a distinct foreign agent, the foreign agent is a default router for the mobile host and is essentially the mobile host's only connection to the network. Because there are two separate routing tables – one on the mobile host, which always points to the foreign agent, and one on the foreign agent, containing true topology information – the conflict between mobility routing and "real" routing disappears, and a simple implementation using a VIF is sufficient but less flexible.

## 6. Future Work

Although we believe our work shows we can make changes in network interfaces and routing transparent to higher-level software, we may not wish to hide changes in the underlying network performance characteristics. Bandwidth, latency, bit error rates, security, and cost can all differ significantly from one type of network to another. We believe it may be advantageous to inform upper-layer network protocols and some applications of these changes so they can adjust their behaviors accordingly. Part of our future work is to investigate what common functionalities should be built into the kernel to cope with these varying quality-of-service parameters, and what application programming interface best enables applications to specify their interests and receive notification of any relevant network changes. Developing a clean interface for this is a major goal of our further work.

As for further work on mobile IP, we plan to experiment with techniques for determining when to switch between networks, and we plan to test some additional routing optimizations described in this paper.

## 7. Conclusions

We have implemented a self-contained mechanism that enables hosts to switch between different networks and network devices without losing connectivity. Our system does not assume the existence of separate foreign agents, allowing our mobile hosts to visit networks that do not support any mobile IP protocols. This enables host mobility over a wide range of networks controlled by different authorities and brings us closer to our goal of ubiquitous connectivity. The performance of our implementation shows that we can switch between devices and IP addresses with minimal disruption to the higher levels of software.

## 8. Source Code Availability

The software for mobile hosts and home agents will be freely available via the Internet. Please use the following URL for details of our project and software release: `http://plastique.stanford.edu/mosquito.html`. We also hope to release our code for DHCP and an extended version of DNS on Linux. Availability of any of this code will also be posted on our web site.

## 9. Acknowledgments

We gratefully acknowledge our many discussions on mobile computing with Bart Miller. His thoughts and advice were enormously useful. We also benefited greatly from discussions with Steve Deering, Mike

Spreitzer, Marvin Theimer, and Lixia Zhang. John Chapin, Stacey Doerr, Bart Miller, Elliot Poger, Mema Roussopoulos, Diane Tang, and Marvin Theimer provided their thoughtful comments on the paper. We thank Metricom for their generosity in loaning us radios and answering many technical questions. We thank Xerox PARC for providing financial support for Xinhua. This work has also been supported by Stanford's Telecom Center and an NSF Faculty Career Award (contract CCR-9501799).

## 10. References

1. Mary Baker, "Changing Communication Environments in MosquitoNet." *Proceedings of the IEEE Workshop on Mobile and Computing Systems and Applications,* December 1994.
2. Trevor Blackwell et al., "Secure Short-Cut Routing for Mobile IP." *1994 Summer USENIX,* June 1994.
3. Stuart Cheshire and Mary Baker, "Experiences with a Wireless Network in MosquitoNet." *Proceedings of the IEEE Hot Interconnects III Symposium on High Performance Interconnects*, August 1995. A version of this paper will also appear in *IEEE Micro*.
4. R. Droms, "Dynamic Host Configuration Protocol." *RFC 1541*, October 1993.
5. Joe Hung, Bart Miller and Mary Baker, CS 244B Course Project, Stanford University Computer Science Department, Spring 1995.
6. John Ioannidis and Gerald Q. Maguire Jr., "The Design and Implementation of a Mobile Internetworking Architecture." *1993 Winter USENIX,* Jan. 1993.
7. David B. Johnson, and Charles Perkins, "Route Optimization in Mobile IP." *Network Working Group, Internet Draft (work in progress),* July 7, 1995.
8. The Linux Journal, P.O. Box 85867, Seattle, Washington, 98145. Editor Phil Hughes, publisher Robert F. Young, 1 (1), March 1994.
9. P. Mockapetris, "Domain Names - Concepts and Facilities." *RFC 1034*, November 1987.
10. C. Perkins, "IP Mobility Support." *Internet Engineering Task Force, Internet Draft (work in progress),* July 8, 1995.
11. Charles E. Perkins and Tangirala Jagannadh, "DHCP for Mobile Networking with TCP/IP." *IEEE ISCC'95*, Alexandria, June 1995.
12. Charles Perkins, and David B. Johnson, "Mobility Support in IPv6." *IPv6 Working Group, Internet Draft (work in progress)*, July 8, 1995.
13. Charles E. Perkins, Andrew Myles, and David B. Johnson, "The Internet Mobile Host Protocol (IMHP)." *Proceedings of INET' 94,* June 1994.
14. M. Pettus, "Unlicensed Radio Using Spread Spectrum: A Technical Overview." Available from Metricom, Inc., Sept. 27, 1993.
15. J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-End arguments in System Design." *ACM Transactions on Computer Systems*, 2 (4), November 1984.
16. Fumio Teraoka, Keisuke Uehara, Hideki Sunahara, and Jun Murai, "VIP: A Protocol Providing Host Mobility." *Communications of the ACM,* Aug. 1994.
17. Mark Weiser, "The Computer for the 21st Century." *Scientific American*, September 1991.

## 11. Biographical Information

**Mary Baker** is an assistant professor in the Departments of Computer Science and Electrical Engineering at Stanford University. Her interests include operating systems, distributed systems, and software fault tolerance. She received her Ph.D. in computer science in 1994 from U. C. Berkeley.

**Xinhua Zhao** is a Ph.D. candidate in the Department of Computer Science at Stanford University. His interests include operating systems and computer networks. Before coming to Stanford, he was a student at the University of Science and Technology in China.

**Stuart Cheshire** is a Ph.D. candidate in the Department of Computer Science at Stanford University. His interests include networks and operating systems. Cheshire received his first class honours degree from Sidney Sussex College, Cambridge in 1989.

**Jonathan Stone** is a Ph.D. candidate in the Department of Computer Science at Stanford University. His interests include networking, distributed systems, and operating systems. He received an M.S. with distinction in computer science in 1991 from Victoria University of Wellington, New Zealand.

The authors' email addresses are
{mgbaker,cheshire,jonathan,zhao}@cs.stanford.edu.

Their postal address is Computer Science Department, Stanford University, Stanford, CA 94305.