To appear in ACM/Baltzer Journal on Special Topics in Mobile Networks and Applications (MONET) (First quarter, 2001)

Flexible Network Support for Mobile Hosts

Xinhua Zhao [a] Claude Castelluccia [b] Mary Baker [a]

[a] *Computer Science Department, Stanford University, Stanford, CA 94305*

[b] *INRIA Rhone-Alpes, 38330 Montbonnot Saint Martin, France*

Fueled by the large number of powerful light-weight portable computers, the expanding availability of wireless networks, and the popularity of the Internet, there is an increasing demand to connect portable computers to the Internet at any time and in any place. However, the dynamic nature of a mobile host's connectivity and its use of multiple network interfaces require more flexible network support than has typically been available for stationary workstations.

This paper introduces two flow-oriented mechanisms, in the context of Mobile IP [23], to ensure a mobile host's robust and efficient communication with other hosts in a changing environment. One mechanism supports multiple packet delivery methods (such as regular IP or Mobile IP) and adaptively selects the most appropriate one to use according to the characteristics of each traffic flow. The other mechanism enables a mobile host to make use of multiple network interfaces simultaneously and to control the selection of the most desirable network interfaces for both outgoing and incoming packets for different traffic flows. We demonstrate the usefulness of these two network layer mechanisms and describe their implementation and performance.

1. Introduction

Light-weight portable computers, the spread of wireless networks and services, and the popularity of the Internet combine to make mobile computing an attractive goal. With these technologies, users should be able to connect to the Internet at any time and in any place, to read email, query databases, retrieve information from the web, or entertain themselves.

To achieve the above goal, a mobile host requires a unique unchanging address, despite the fact that as it switches from one communication medium to another, or from one network segment to another, the IP address associated with its network interface must change accordingly. The IP address must change because IP [25] assumes that a host's IP address uniquely identifies the segment of the network through which a host is attached to the Internet. Unfortunately, changing the address will break ongoing network conversations between a mobile host and other hosts, because connection-oriented protocols such as TCP [26] use the IP addresses of both ends of a connection to identify the connection. Therefore, Mobile IP [23], or another similar mechanism that allows a host to be addressed by a single address, is needed to accommodate host mobility within the Internet.

However, due to the dynamic nature of a mobile host's connectivity and its use of multiple interfaces, providing network support for a mobile host can be a much more complex task than for its stationary counterparts. Mobile IP takes an important step towards supporting mobility and is the context for our work, but through day-to-day experience using the MosquitoNet [16] mobile network, we have found that it is desirable to have finer control over the network traffic of a mobile host on a per flow basis than is provided by Mobile IP. The work presented in this paper is mainly motivated by the following observations on the unique characteristics of a mobile host:

- Duality: The mobile host has two roles as both a host virtually connected to its home network and as a normal host on the network it is visiting. In this sense, we can consider the mobile host to be virtually multi-homed. This duality brings along increased complexity as well as opportunities for optimization.

- Dynamically changing point of attachment: We connect a portable to various networks, such as our office network while at work, the network of a wireless Internet access service provider while on the road, or another network at home or elsewhere. Different networks may have different policies for dealing with packets from mobile hosts. Depending on where a mobile host currently operates and with whom it communicates, it may need to use different packet delivery methods that are either more robust or more efficient.
- Multiple network interfaces: To achieve connectivity in any place at any time, mobile hosts will likely require more than one type of network device. For example, our mobile hosts use Ethernet or WaveLAN [28] when in a suitably equipped office or home, but they use a slower wireless packet radio network such as Metricom Ricochet [18] elsewhere.
  There is no single network device that can provide the desired quality of service (QoS) all the time. There is always a trade-off among coverage, performance, and price. There are even times when multiple network devices need to be used at the same time (such as a satellite connection for downlink and a modem connection for uplink). In this case, the mobile host is physically multi-homed as well. Making use of these network interfaces simultaneously for different flows of traffic is a challenge.

Our goal is to enable a mobile host to communicate both robustly and efficiently with other hosts as it moves from place to place. While there are ways to satisfy one goal or the other, satisfying both at once is a challenge. For example, we can treat a mobile host as virtually connected to its home network by always tunneling packets between the mobile host and its home agent. Although robust, this is obviously not efficient. Achieving efficiency as well requires a mobile host to have more flexibility than has been provided by previously existing mechanisms.

This paper addresses the following two issues in providing the flexibility desirable for a mobile host. First is the need at the network routing layer to support multiple packet delivery methods (such as whether to use regular IP or Mobile IP). At the network layer, we have developed a general-purpose mechanism, the Mobile Policy Table, which supports multiple packet delivery methods simultaneously for different flows and adaptively selects the most appropriate method according to the characteristics of each traffic flow. Second is the need to make use of multiple network interfaces simultaneously and to control the interface selection of both outgoing and incoming packets for different traffic flows. This is achieved by extending the base Mobile IP protocol to control the choice of interfaces to use for incoming traffic to a mobile host. We also amend the routing table lookup to enable the use of multiple network interfaces for outgoing traffic flows from a mobile host. The result is a system that applies Mobile IP more flexibly by taking into consideration traffic characteristics on a flow-by-flow basis.

The rest of the paper is organized as follows: In section 2, we give a brief description of Mobile IP, the context in which our work is done. In section 3, we illustrate scenarios for the use of multiple packet delivery methods for different flows of traffic. In section 4, we detail the general-purpose mechanism that supports this use of multiple packet delivery methods. In section 5, we describe the simultaneous use of multiple network interfaces. In section 6, we report the implementation status of the system and present the results of system performance measurements obtained from our experiments. In section 7, we list related work. In section 8, we consider the applicability and potential of our work with IPv6. Finally, we present conclusions together with some future and continuing work in section 9.

2. Background: Mobile IP

Mobile IP [23] is a mechanism for maintaining transparent network connectivity to mobile hosts. Mobile IP allows a mobile host to be addressed by the IP address it uses in its home network (home IP address), regardless of the network to which it is currently physically attached. Figure 1 illustrates the operation of basic Mobile IP.

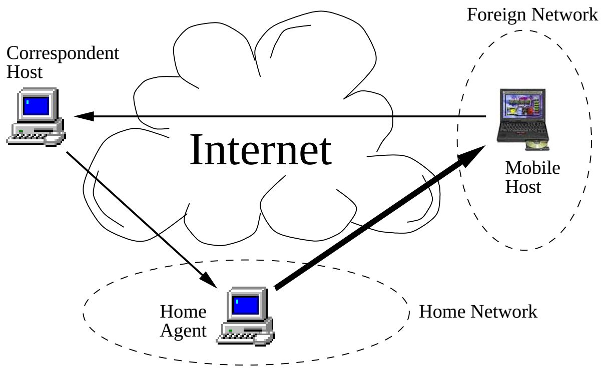

Figure 1. Basic Mobile IP Protocol. Packets from the correspondent host to the mobile host are always sent to the mobile host's home network first, and then forwarded (tunneled) by the home agent to the mobile host's current point of attachment (care-of address). Packets originating from the mobile host are sent directly to the correspondent host, thus forming a triangular route. The thick line indicates that the original packet is encapsulated in another IP packet when forwarded, and is therefore of a larger size.

The Mobile IP specification allows for two types of attachment for a mobile host visiting a "foreign" network (a network other than the mobile host's home network). For the first type of attachment, the mobile host can connect to the foreign network through a "foreign agent", an agent present in the foreign network, by registering the foreign agent's IP address with its home agent. The home agent then tunnels [22] packets to the foreign agent, which decapsulates them and sends them to the mobile host via link-level mechanisms.

Although our Mobile IP implementation supports the use of foreign agents, our work is more focused on the second type of attachment, which provides a mobile host with its own "co-located" care-of address in the foreign network. In this scenario, the mobile host receives an IP address to use while it visits the network, via DHCP [6] or some other protocol or policy. It registers this address with its home agent, which then tunnels packets directly to the mobile host at this address. The disadvantage of this scenario is that the mobile host has to decapsulate packets itself and more IP addresses are needed in the foreign network, one for each visiting mobile host. The advantage is that the mobile host also becomes more directly responsible for the addressing and routing decisions for the packets it sends out, and it therefore has more control over such operations.

While Mobile IP has laid the groundwork for Internet mobility, there are still many challenges to tackle, as seen from on-going efforts in this area. These efforts include route optimization [13], firewall traversal [20], and "bi-directional tunneling" (or "reverse-tunneling") [19] to allow packets to cross security-conscious boundary routers. This last problem, as described in section 3.2, is one of our motivations for making it possible for mobile hosts to choose dynamically between different packet addressing and routing options. We believe that these efforts make evident the inherent need for mobile hosts to use different techniques under different circumstances.

3. Supporting Multiple Packet Delivery Methods

Supporting multiple packet delivery methods is the first area in which we have increased the flexibility of mobile hosts. By avoiding a single method of delivery, the mobile host only pays for the extra cost of mobility support or security perimeter traversal when it is truly needed.

We illustrate some of the situations for which we have found such flexibility to be beneficial in practice. Although Figure 2 only shows the examples we have implemented so far, the mechanism we propose here can be extended to support other delivery methods when other choices become desirable.

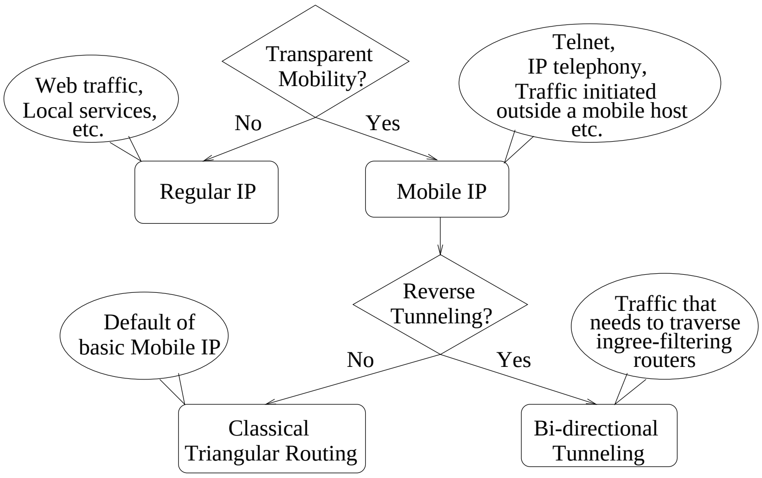

Figure 2. Simultaneous Use of Multiple Packet Delivery Methods. This figure summarizes the different packet delivery choices (in rectangle boxes) we have implemented so far. Listed in the circles are example uses for each packet delivery choice. The policy decisions to be made are in diamonds.

3.1. Providing Transparent Mobility Support Only When Necessary

The transparent mobility support of Mobile IP is important for long-lived connection-oriented traffic or for incoming traffic initiated by correspondent hosts. However, this transparent mobility support does not come without cost. In the absence of route optimization for Mobile IP, packets destined to a mobile host are delivered to its home network and then forwarded to the mobile host's current care-of address in the network it is visiting. If a mobile host is far away from home but relatively close to its correspondent host, the path traversed by these packets is significantly longer than the path traveled if the mobile host and the correspondent host talk to each other directly. The extra path length not only increases latency but also generates extra load on the Internet. It even increases load on the home agent, potentially contributing to a communication bottleneck if the home agent is serving many mobile hosts simultaneously.

It would be ideal to use Mobile IP route optimization [13]. However, since Mobile IP route optimization requires extra support on correspondent hosts in addition to support on the mobile host and its home agent, it requires widespread changes throughout the Internet, which is unrealistic for the near future.

Fortunately, there are certain types of traffic for which a mobile host may not require Mobile IP support. Examples are most web browsing traffic and communication with local services discovered by the mobile host. Web connections are usually short-lived, so it is unlikely that a mobile host will change its foreign network address in the midst of a connection (exceptions are web push technology and HTTP 1.1 [8], which can potentially use persistent transport connections). Even if it does, the user can simply press reload, and the web transfer will be retried. With such traffic, we can avoid the extra cost associated with mobility support.

3.2. Supporting Bi-directional Tunnels and Triangle Routes

When communication requires transparent mobility, there still remains a choice of packet delivery methods. An important example is communication that must traverse security-conscious boundary routers.

As a result of IP address spoofing attacks and in accordance with a CERT advisory [4], more routers are filtering on the source address (ingress filtering) [7] and will drop a packet whose address is not "topologically correct" (whose originating network cannot be the one identified by the source address). In the presence of such routers, the triangle route as specified in the basic Mobile IP protocol will fail. Figure 3 illustrates an example of this problem.

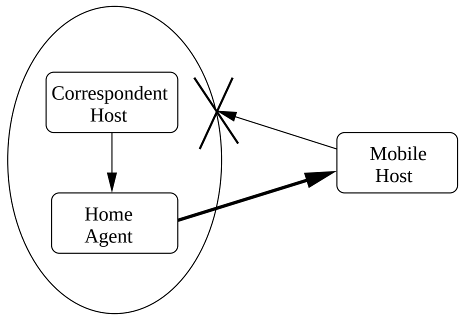

Figure 3. The Problem with Source IP Address Filtering at a Security-conscious Boundary Router. When the mobile host sends packets directly to the correspondent host in its home domain with the source IP address of the packets set to the mobile host's home IP address, these packets will be dropped by the boundary router, because they arrive from outside of the institution and yet claim to originate from within.

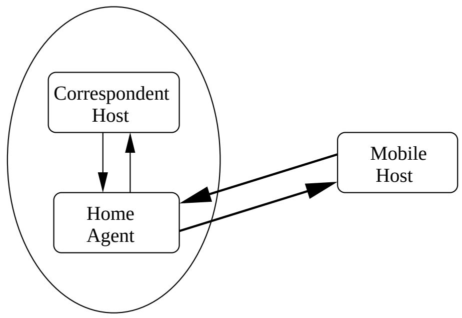

Figure 4. Solution to the Ingress Filtering Problem. To address the problem caused by source IP address filtering on security-conscious boundary routers, the mobile host sends packets by tunneling along the reverse path as well. Since the encapsulated packets in the tunnel from the mobile host to its home agent use the mobile host's care-of address as their source IP address (which is topologically correct), these packets will no longer be dropped by the security-conscious boundary routers.

As another example, if the boundary router is in the domain visited by the mobile host, it may drop packets that are received from inside but claim to originate from outside; these packets look as if they are "transit traffic", and not all networks will carry transit traffic.

To address the above problems, we can tunnel packets sent by the mobile host through its home agent to its correspondent hosts, in much the same way we tunnel packets sent to the mobile host. This is called "bi-directional tunneling" [19]. Figure 4 illustrates the solution.

This bi-directional tunneling addresses the problem related with ingress filtering routers, but again, this comes with increased cost. If a mobile host visits a network far away from home and tries to talk to a correspondent host in a nearby network, packets originating from the mobile host will now have to travel all the way home and then back to the correspondent host, increasing the length of this reverse path.

However, not all the packets need to be sent this way. It is unnecessary to force all traffic through a bi-directional tunnel just because some ingress filtering routers would drop traffic sent to specific destinations. Such tunneling may be unnecessary for a large part of the traffic for which the topologically incorrect source IP address in packet headers is not a problem. Therefore, it is important to support the use of both triangle routing and bi-directional tunneling simultaneously, so that only those traffic flows that truly need to use bi-directional tunneling pay for this extra cost.

3.3. Joining Multicast Groups in Different Ways

The final packet delivery mechanism we have experimented with is to allow a mobile host to choose to join multicast groups either remotely (through its home network), using its home IP address, or locally, using its co-located care-of address in the foreign network. This flexibility is necessary because multicast traffic is often limited to a particular site by scoping [5]. Also, there are advantages and disadvantages to either choice [1] even if scoping is not an issue.

- Joining locally: In this respect the mobile host is no different from any normal host on the same subnet. The advantage is that the delivery of multicast traffic to the mobile host is more efficient. The disadvantages are that it requires the existence of a multicast-capable router in the foreign network, and those mobile hosts actively participating in the multicast session will have to re-identify themselves within the group when they move to another network.
- Joining through the home network: All multicast traffic has to be tunneled bi-directionally between the mobile host and its home agent. The advantages of this choice are that it does not require multicast support on the foreign network, and the mobile host will retain its membership as it moves around. The disadvantage is that the route is less efficient. Although there are optimizations that allow a single copy of multiple packets to be tunneled to a foreign agent serving multiple mobile hosts [9], the home agent still must tunnel a copy of a multicast packet to each foreign network that has one of its mobile hosts visiting.

We can choose either option for different multicast groups. To join a multicast group locally, we add an entry in the Mobile Policy Table instructing traffic destined to certain multicast addresses to use conventional IP support. To join a multicast group remotely, we add an entry in the Mobile Policy Table to use bi-directional tunneling for traffic destined to these multicast groups. The mobile host also needs to notify its home agent, in a registration packet, to forward multicast traffic to it.

4. A General-purpose Mechanism for Flexible Routing

In this section, we describe the mechanism used to achieve the flexibility features described in the previous section. This general-purpose mechanism is centered around the idea of introducing a Mobile Policy Table in the routing layer of the network software stack. The IP route lookup routine `ip_rt_route` is augmented to take the Mobile Policy Table into consideration together with the normal routing table in determining how a packet should be sent.

4.1. Support in the Network Layer

We choose to add our support for multiple packet delivery methods to the network layer due to its unique position in the network software stack. By modifying the layer through which packets converge and then diverge, we avoid a proliferation of modifications. Relatively fewer changes need to be made in the kernel network software, and both the upper layer protocol modules (such as TCP or UDP) and lower layer drivers for different network devices remain unchanged.

We further identify the route lookup routine as a natural place to add such support. Along with normal route selection, the enhanced route lookup also makes decisions for choosing among multiple ways to deliver packets, as necessitated by the changing environment of a mobile host.

4.2. The Mobile Policy Table

The Mobile Policy Table specifies how the packets should be sent and received for each traffic flow matching certain characteristics. The routing and addressing policy decisions currently supported in the Mobile Policy Table are:

- whether to use transparent mobility support (Mobile IP) or regular IP;
- whether to use triangular routing or bi-directional tunneling, if using Mobile IP.

Destination	Netmask	PortNum	Mobility	Tunneling
171.64.0.0	255.255.0.0	0	Yes	Yes
0.0.0.0	0.0.0.0	80	No	N/A
0.0.0.0	0.0.0.0	0	Yes	No

Table 1
A Sample Mobile Policy Table. This mobile policy table specifies that all traffic destined back to the mobile host's home domain should use bi-directional tunneling, to satisfy the boundary routers at its home institution (first row); all traffic to port 80 (web traffic) should avoid using transparent mobility support (second row); and the remaining traffic should by default use Mobile IP with a regular triangular route (third row). The second entry applies to all traffic with a destination port number of 80, even for destinations matching the first entry, since port number specification takes precedence.

These policies are specified through two types of entries: "per-socket" entries and "generic" entries, with per-socket entries taking precedence. While the Mobile Policy Table only contains generic entries, a per-socket entry kept within the socket data structure allows any application to override the general rules. Without a per-socket entry, traffic is subject only to generic entries in the policy table, which specify the delivery policy for all traffic matching the given characteristics.

For generic entries, the Mobile Policy Table lookup currently determines which policy entry to use based on two traffic characteristics: the correspondent address and port number (for TCP and UDP). The correspondent address is useful, because we often want to treat flows to different destinations differently. The port number is useful as well, because there are many reserved port numbers that indicate the nature of the traffic, such as TCP port 23 for telnet, or port 80 for HTTP traffic. While these are the characteristics currently taken into consideration, we can extend the technique to include other characteristics in the future.

Table 1 shows, as an example, the Mobile Policy Table currently used on our mobile hosts when visiting places outside of their home domain. The Mobile Policy Table lookup operation always chooses the most specifically matched entries (those with more restricted netmask and/or port number specifications) over more general ones.

4.3. How it Works

Figure 5 illustrates the use of the Mobile Policy Table and routing table within the Linux kernel. The modification to the kernel is mainly limited to the route lookup function. For backward compatibility, the normal routing table remains intact. During a route lookup, extra arguments such as other characteristics of the traffic flow (currently only the TCP or UDP port number) and the source IP address chosen are used in addition to the correspondent host's address for deciding how the packet should be sent.

The new route lookup function uses the specified source IP address to determine if the packet is subject to policy decisions in the Mobile Policy Table. If the source IP address has already been set to the IP address associated with one of the physical network interfaces, this indicates that no mobility decision should be made for the packet. Packets may have their source address set either after they are looped back by the virtual interface (described below), since a mobility decision has already been made by that time, or by certain applications. An example of such an application is the mobile host daemon handling registration and deregistration with the home agent; this daemon needs to force a packet through a particular real interface using regular IP. In cases where the source IP address is already set, only the normal routing table is consulted based upon the destination address, and the resulting route entry is returned. For the rest of the packets, the Mobile Policy Table needs to be consulted to choose among multiple packet delivery methods.

The virtual interface ("*vif*") handles packets that need to be encapsulated and tunneled. It provides the illusion that the mobile host is still in its home network. Packets sent through *vif* are encapsulated and then looped back to the IP layer (as shown by the wide bi-directional arc in the figure) for delivery to the home agent. This time, however, the source IP address of the encapsulating packet has already been chosen,

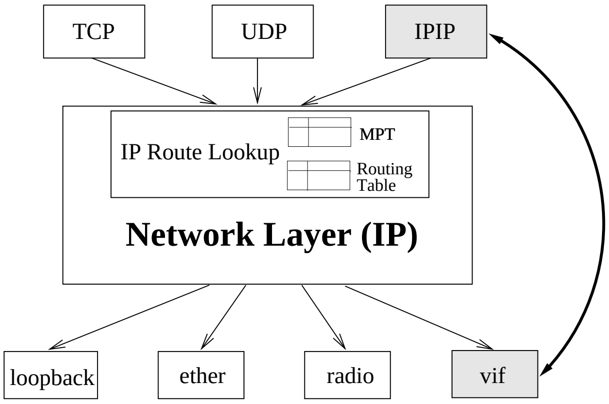

Figure 5. This figure shows where the Mobile Policy Table (MPT) fits into the link, network, and transport layers of our protocol stack. The MPT resides in the middle (network) layer, consulted by the IP route lookup function in conjunction with the normal routing table to determine how packets should be sent. The bottom (link) layer shows the device interfaces, with *vif* being a virtual interface that handles encapsulation and tunneling of packets. The top (transport) layer shows TCP and UDP, along with an IP-within-IP processing module. The arrows depict the passing of data packets down the layers. The shaded boxes indicate that the IPIP and *vif* modules have overlapping functionalities. The bold bi-directional arc shows the need to pass packets between the two modules.

so it will now be sent through one of the physical interfaces. Accordingly, packets being tunneled to the mobile host by its home agent are also looped by the IPIP module to *vif*, so that they appear to have arrived at the mobile host as if it were connected to its home network.

To maintain reasonable processing overhead, policy table entries are cached in a manner similar to routing table entries. If the characteristics of the traffic match a cached entry, the software uses the cached entry to speed up the process of policy lookup. Otherwise, a new policy table lookup will be carried out. Whenever the mobile policy table is modified, the cached entries are flushed.

We believe this is a general-purpose mechanism, because it can be easily extended to take other traffic characteristics into consideration and to add more policy decisions if it becomes desirable to do so. For instance, if particular correspondent hosts have the ability to decapsulate packets themselves, we could note this information in the Mobile Policy Table for those destinations, and the mobile host could send encapsulated packets directly to these hosts, bypassing the home agent yet still providing the robustness of a bi-directional tunnel.

4.4. Dynamic Adaptation of the Mobile Policy Table

The entries in the Mobile Policy Table can be changed at runtime to specify a different policy. We currently have the following two mechanisms to adjust entries in the Mobile Policy Table dynamically.

First, we can swap in a new set of entries for the Mobile Policy Table whenever a mobile host changes its care-of address. When a mobile host changes its point of attachment to the Internet, the best way to communicate with other hosts often changes accordingly. For example, when our mobile hosts move from within the Stanford domain to a foreign network outside, they need to switch to using reverse tunneling for traffic back to their home domain due to the ingress filtering routers present at the boundary of Stanford's domain. For foreign networks frequently visited by a mobile host, we use predefined configuration files that contain the set of entries to use in the Mobile Policy Table so that a suitable set of policies can be put in place quickly when a mobile host changes location. For previously unknown foreign networks, a default set of policies will be used.

In addition, we make it possible to change specific entries in the Mobile Policy Table dynamically. For instance, we support adaptively selecting between the most robust packet delivery method (i.e. bi-directional tunneling) and a more efficient packet delivery method (i.e. triangular routing). This mechanism helps when a current setting in the Mobile Policy Table fails or when it is simply impossible to specify the right policy beforehand. The adaptation applies only to cached entries, i.e. we only adjust those entries that are actually in use.

To implement this mechanism, we automatically dispatch a separate process that probes each destination address by sending ICMP echo requests using triangular routing, with the interval between probes being increased exponentially up to a preconfigured maximum value. For a flow using reverse tunneling, if a reply to any of the probes is successfully received, the Mobile Policy Table entry for this flow will be changed from bi-directional tunneling to use the more efficient triangular routing. For a flow using triangular routing, the Mobile Policy Table entry is changed back to use bi-directional tunneling if a certain number of probes (currently, five) have failed in a row. After the initial stage, this separate process keeps probing in the background with the maximum probe interval. If a series of probes fails while a flow is using triangular routing, we switch to use bi-directional tunneling. If a probe succeeds while a flow is using bi-directional tunneling, the Mobile Policy Table entry will be changed to use triangular routing immediately. In all other cases, no action is necessary. This way, a flow is able to select adaptively the most efficient packet delivery method.

5. Supporting the Use of Multiple Network Interfaces Simultaneously

The use of multiple network interfaces is different on mobile hosts than on typical stationary machines. Stationary hosts with multiple network interfaces are usually routers, forwarding packets with different destination addresses through different network interfaces. On a mobile host, these network devices instead represent different ways this single mobile host can communicate with the outside world. For example, we may want to use two different interfaces (one for telnet and another for file downloading) for communication with the same host.

Although some operating systems can direct broadcast or multicast traffic out through a particular network interface to its immediately connected subnets, all these operating systems (except for Linux) make routing decisions by destination addresses alone, without taking other traffic flow characteristics into consideration. The standard Linux distribution includes our contribution that enables each socket to choose among different simultaneously active network interfaces for outgoing traffic to destinations beyond the directly connected subnets. Our goal is to enable a mobile host to make use of multiple active network interfaces simultaneously for different flows of traffic in a more general way.

5.1. Motivation for Multiple Interfaces

We find the support for the use of multiple active interfaces useful for the following reasons:

1. Smoother hand-offs: With the ability to use multiple network interfaces simultaneously, a mobile host can probe the usability of other interfaces beyond its directly connected subnet without disturbing the interface currently in use. The current network interface can remain in use until the new care-of address on the new network can be successfully registered with the home agent. This eliminates unnecessary packet loss when switching care-of addresses.

2. Quality of service (QoS): The different physical networks to which a user has access may offer different QoS guarantees. For example, a mobile user may have simultaneous access to a GSM data network [27] that has low bandwidth but relatively low latency, as well as to a Metricom Ricochet network that offers higher bandwidth but has higher and more variable latency. The mobile host might decide to use the GSM network for its low-bandwidth interactive flows (such as its telnet traffic) which require low latency for user satisfaction, while using the Metricom network for its bulk data transfer flows (such as ftp traffic) which require high bandwidth but do not demand as low a latency. The combination of these different types of networks can deliver a larger range of QoS to mobile users.

3. Link asymmetry: Some networks only provide unidirectional connectivity. This is the case for many satellite systems, which usually provide downlinks only. In these systems, connectivity in the reverse direction is provided via a different means, such as a SLIP or PPP dialup line, a cellular modem, or a CDPD [2] device. Being able to specify the incoming and outgoing interfaces explicitly in a natural manner is a useful feature.
4. Cost and billing: The cost of accessing different networks may be a decisive factor in interface selection. Users might select different interfaces according to the identity of the bill payer. For example, they might choose a cheaper and lower quality access network for personal communications and a more expensive and better quality one for business communications.
5. Privacy and security: Privacy and security may also be of considerable importance in interface selection. Users may trust some networks more than others and prefer to use them for confidential and important communications. They might also want, for privacy reasons, to choose different networks for business and personal communications.

The following sections explain how to support this feature for packet transmission and reception.

5.2. Supporting Multiple Interfaces in Transmission

For a mobile host to use multiple network interfaces simultaneously to send packets, we have devised the following two techniques. The first is to make use of the metric field in the existing routing table entry so that multiple routes through different interfaces can be associated with a certain destination specification. Usually the normal default route has a metric of one. Routes through other interfaces with metrics greater than one can coexist with the normal default route in the routing table. A route lookup that does not specify a particular interface will thus find the normal default route, maintaining backwards compatibility, since the lookup always chooses the matching route with the smallest metric.

The second technique we provide is a "bind-to-device" socket option that applications can call to associate a specific device with a certain socket. The route lookup routine has been modified so that when a device selection is specified, only those route entries associated with the particular device will be considered. Therefore, different applications running simultaneously can each choose different network interfaces to use for sending packets.

5.3. Supporting Multiple Interfaces in Reception

5.3.1. Overview of the scheme

A mobile host with multiple interfaces running standard Mobile IP cannot receive different flows of packets on different interfaces simultaneously. With Mobile IP, a mobile host sends location updates that indicate its current ⟨ *home address, care-of address* ⟩ binding to its home agent and possibly to its correspondent hosts. As a result, all packets addressed to a mobile host are sent over the same interface or possibly over multiple interfaces all at the same time if the "S" (simultaneous binding) flag is set in the registration.

We have extended Mobile IP to allow a mobile host to use different interfaces to receive different flows. In our work, we define a flow as a triplet: ⟨ *the mobile host's home IP address, the correspondent host's IP address, the port number on the correspondent host* ⟩. Our definition is different from typical flow definitions in that we do not include certain fields such as the port number on the mobile host. This actually makes certain flows from the same mobile host indistinguishable from each other. However, since our current goal is to treat traffic flows differently based on whom a mobile host is talking to and the nature of the communication (for example, whether it is interactive traffic or bulk transfer), we believe that this triplet is sufficient in capturing the essential traffic flow characteristics to serve this purpose.

In our framework, a mobile host may choose to receive the packets belonging to a given flow on any of its interfaces by sending a Flow-to-Interface binding to its home agent. This Flow-to-Interface binding

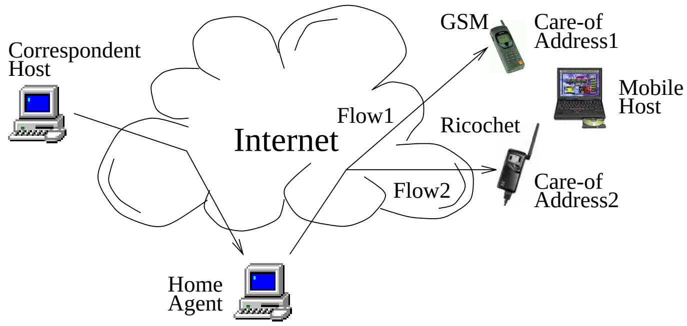

Figure 6. Supporting Multiple Incoming Interfaces. Packets belonging to any flows addressed to the mobile host arrive on the home network via standard IP routing. The home agent intercepts packets and tunnels them to the care-of address selected based on the packets' destination addresses, source addresses and source ports. Packets of flow1 are tunneled to Care-of Address1, while packets of flow2 are tunneled to Care-of Address2.

specifies the mobile host's care-of address that the home agent should use to forward packets belonging to the flow.

Upon reception of a Flow-to-Interface binding, a home agent updates its extended binding list, which contains entries that associate a particular flow specification with a mobile host's care-of address. Thereafter, when the home agent receives a packet addressed to a mobile host it is serving, it searches in its extended binding list for an entry matching the corresponding fields of the packet and forwards the packet to the associated care-of address. If no entry is found, the packet is forwarded to the mobile host's default care-of address.

Figure 6 illustrates the routing of datagrams to a mobile host away from home, once the mobile host has registered several Flow-to-Interface bindings with its home agent. It is worth noting that since the port number on a correspondent host is a factor in distinguishing between different flows, the home agent must treat already fragmented packets to mobile hosts specially. Currently, we reassemble these fragmented packets at the home agent before forwarding them to mobile hosts. Since all packets pass through the home agent, we do not suffer from the problem normally associated with doing reassembly at intermediate routers (wherein the routers are not guaranteed to see all the fragments). As we describe in Section 8, the use of IPv6 flow label will eliminate the need for de-fragmentation.

The Flow-to-Interface bindings are registered to the home agent via two new extensions of the Mobile IP registration messages. The following sections review briefly the Mobile IP registration procedure and detail the new extensions we have devised.

5.3.2. The Mobile IP Registration Procedure

A mobile host and its home agent exchange Mobile IP registration request and reply messages in UDP packets. The format of the registration request packet is shown in Figure 7. The fixed portion of the registration request is followed by extensions, a general mechanism to allow optional information and protocol extensibility. We use this general mechanism to extend the Mobile IP protocol so that different flows to a mobile host can be associated with different network interfaces on that host.

5.3.3. The Flow-to-Interface Binding Extension

A mobile host may ask a home agent to register a number of Flow-to-Interface bindings by appending to a registration request a list of Flow-to-Interface extensions, each defining a Flow-to-Interface binding to be registered. The Flow-to-Interface binding extension format is shown in Figure 8.

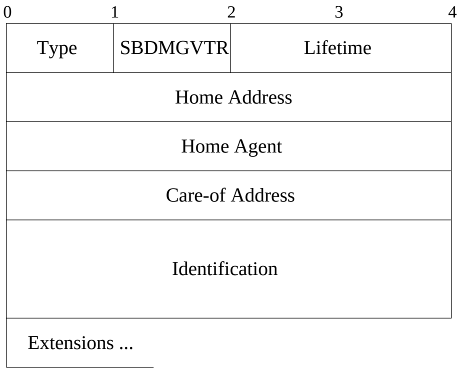

Figure 7. Mobile IP Registration Request. A registration request is used by a mobile host to create, on its home agent, a mobility binding from the static IP address at its home network (i.e. its home address) to its current care-of address. A general extension mechanism is defined for extending the protocol.

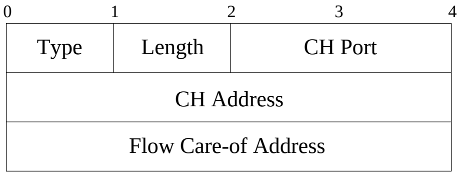

Figure 8. Flow-to-Interface Binding Extension. The CH (correspondent host) Address and the CH Port fields in this extension together with the mobile host's home address define the flow that should now be associated with the care-of address specified in the Flow Care-of Address field.

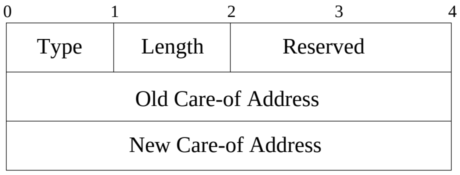

Figure 9. Flow-to-Interface Binding Update Extension. All flows previously bound to the old care-of address should now be redirected to the new care-of address.

5.3.4. The Flow-to-Interface Binding Update Extension

A mobile host may ask a home agent to redirect certain existing flow bindings to a different care-of address by using the Flow-to-Interface binding update extension. The binding update extension format is detailed in Figure 9. This extension is useful when a mobile host changes the point of attachment of one of its interfaces and obtains a new care-of address for this interface.

5.3.5. Some Compatibility Issues

We want to ensure that unmodified mobile hosts are able to use our enhanced home agent, and that our enhanced mobile hosts work properly with unmodified home agents. The first scenario is not an issue, since the enhanced home agent can process both regular registration messages as well as those with our new extensions. However, the second scenario requires further consideration.

According to the Mobile IP specification [23], when an extension numbered within the range 0 through 127 is encountered but not recognized, the message containing that extension must be silently discarded. When an extension numbered in the range 128 through 255 is encountered but not recognized, only that particular extension is ignored, but the rest of the extensions and the message must still be processed. We choose to number our new extensions within the range 128 to 255, since it is undesirable to have registration packets silently discarded, causing the system to wait for timeouts instead.

When a mobile host sends a Flow-to-Interface binding registration to a home agent that does not support Flow-to-Interface bindings, the packets will still be processed. This is safe, since the registration message is a normal registration message excluding the Flow-to-Interface binding extension. However, to distinguish these unsuspecting home agents from the enhanced home agents, we make the enhanced home agents use different return codes in reply to registration requests with Flow-to-Interface extensions. If the registration is accepted, the home agent returns the code 2 instead of 0. The return code 3 (instead of 1) will be used if the registration is accepted and simultaneous mobility binding is granted. Therefore, if a mobile host sends out registration requests with Flow-to-Interface binding extensions but gets a return code 0 or 1 in reply, it should stop sending Flow-to-Interface binding extensions to its home agent, since continuing to send them will just waste bandwidth.

6. Implementation Status and Experiments

6.1. Implementation Status

We have implemented the support for multiple packet delivery methods and simultaneous use of multiple network interfaces on top of our Mobile IP implementation [3] under Linux (currently kernel version 2.0.36). Some core functions of our Mobile IP implementation reside within the kernel, such as packet encapsulation and forwarding. Other functions, such as sending and receiving registration messages, are implemented in a user-level daemon.

A mobile host can choose the use of either Mobile IP or regular IP and the use of either bi-directional tunneling or triangular routing for different flows of traffic all at the same time. We also provide mobility-aware applications with the flexibility to choose incoming and outgoing interfaces. We use two new socket options to bind flows to given interfaces:

- SO_BINDTODEVICE: [1] This option is used by an application to bind the outgoing flows of a socket to an interface.
- SO_BIND_FLOWTODEVICE: This option is used by an application to bind the incoming flows of a socket to a given interface.

6.2. Experiments with Multiple Packet Delivery Methods

6.2.1. Benefits

The choice of delivery method has a performance impact on traffic flows. We look at a mobile host in one particular real scenario to illustrate the potential benefits certain flows will be able to receive when they can use the more efficient delivery methods made possible by our flexible mechanism. Note that other scenarios could see very different results depending on the actual network latency between the mobile host, the home agent and the correspondent host.

We evaluate the latency improvement resulting from using the most direct route possible under the circumstances. For these experiments, the mobile host connects to a foreign network in one of our campus residences, which is also on Ethernet, and registers its current care-of address with its home agent. The setup of the test scenario is illustrated in Figure 10. The mobile host sends ping (ICMP echo request) packets to the default gateway (acting as its correspondent host) of its local subnet and we measure the round-trip latency using the following three delivery methods, switching between them by manipulating the Mobile Policy Table:

- Regular IP (no transparent mobility support);

[1] The name SO_BIND_OUTFLOW_TO_INTERFACE would be more appropriate, and the next socket option should also be named accordingly. Unfortunately, our original name SO_BINDTODEVICE is already in the standard Linux distribution.

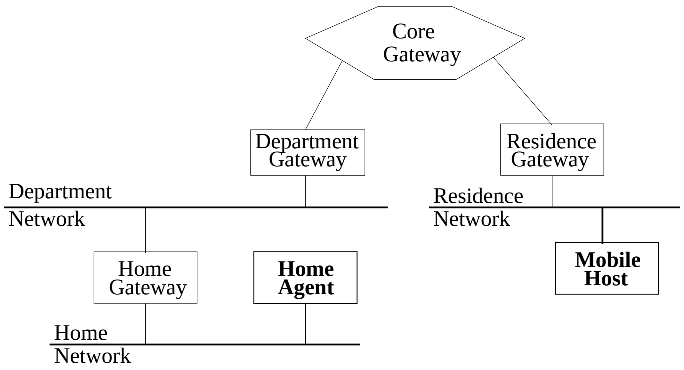

Figure 10. Setup of the Test Environment for Experiments on Multiple Packet Delivery Methods. The mobile host visits the campus residence network as a foreign network. The mobile host is a Thinkpad 560, and the home agent is a 90MHz Pentium. They are both running Linux.

Delivery Method	Min (ms)	Max (ms)	Avg (ms)	Standard Deviation
Bi-directional Tunneling	7.3	9.3	7.9	0.36
Triangular Route	4.5	6.2	5.0	0.35
Regular IP	2.0	4.0	2.4	0.35

Table 2
Latency Comparison When Using Small Packets. Using 64 bytes of ICMP data (i.e., the default ping packet size), the test for each delivery method is repeated 100 times with an interval of 2 seconds.

Delivery Method	Min (ms)	Max (ms)	Avg (ms)	Standard Deviation
Bi-directional Tunneling	37.4	42.4	38.4	1.2
Triangular Route	24.6	30.9	25.4	1.0
Regular IP	11.5	14.5	12.4	0.6

Table 3
Latency Comparison When Using Large Packets. Using 1440 bytes of ICMP data, the test for each delivery method is repeated 100 times with an interval of 2 seconds.

- Triangular delivery (the default behavior in Mobile IP);
- Bi-directional tunneling (for security-conscious boundary routers).

We collect the data by repeating the above test 100 times with an interval of two seconds for each delivery method. The results are shown in Table 2 for small packets and Table 3 for large packets. For this experimental setup, our flexible mechanism reduces the latency by one half for both small and large packets when choosing regular IP over the default Mobile IP behavior. Even when mobility support is necessary, this flexible mechanism still reduces the latency by about one third for small and large packets when we can choose the triangular route over the robust but more costly bi-directional tunnel.

6.2.2. Cost

We measure both the time spent in doing regular route lookup and the time spent in doing route lookup with a Mobile Policy Table lookup on a mobile host. We then examine the difference to determine the added latency in making policy decisions according to the Mobile Policy Table.

The results are shown in Table 4. The overhead of consulting the Mobile Policy Table in route lookups is small, less than 20 μs even when the entries are not currently cached. This is less than one percent of the

Experiment	Cached Time (μs)	Cached Standard Deviation	Uncached Time (μs)	Uncached Standard Deviation
Regular route lookup	18.5	0.8	100.1	0.9
Route lookup w/ MPT	23.5	0.7	116.5	1.2

Table 4

Cost to Support the Simultaneous Use of Multiple Packet Delivery Methods. This experiment measures both the time needed for the regular route lookup and the time spent to do the modified route lookup with Mobile Policy Table (MPT) consultation. Each measurement is repeated 10 times. We tested cases for both a cache hit and a cache miss when looking up routes.

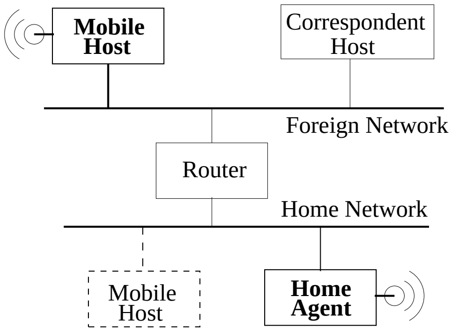

Figure 11. Setup of the Test Environment for Experiments on Flow Demultiplexing. The Mobile Host is a Gateway2000 (486DX2-40 processor) and the Home Agent is a Toshiba Tecra (Pentium 133MHz), both of which have an Ethernet interface as well as a Metricom radio interface. The Correspondent Host is a 90MHz Pentium. All machines are running Linux.

round-trip time between two hosts on the same Ethernet segment (typically around 2ms).

6.3. Experiments with Multiple Active Interfaces

The main possible source of overhead for supporting multiple interfaces is the flow demultiplexing processing on the home agent, which may affect both the latency and the throughput. The goal of the experiments described in this section is to evaluate the latency cost of this demultiplexing, i.e. the extra time it takes for a home agent to forward a packet addressed to a mobile host.

The setup of our test environment is illustrated in Figure 11. For these experiments, the mobile host connects to a foreign network on one of our department's Ethernets and registers its current care-of address with its home agent. The correspondent host sends *ping* packets to the mobile host. We measure the flow binding demultiplexing time at the home agent by monitoring the code using the Linux `do_gettimeofday` kernel function. We consider two cases: when the home agent has no Flow-to-Interface bindings, regular Mobile IP is used; when the home agent contains Flow-to-Interface bindings, it must search the list of bindings before forwarding a packet to the mobile host.

We collect the data by repeating the above test 10 times each as the number of bindings in the home agent's list increases from 0 to 60. Table 5 displays the results.

These results for the flow binding demultiplexing cost may seem large compared to the demultiplexing time incurred with regular Mobile IP on the home agent (2.1 μs), but the extra cost is a small additional factor when compared to the round-trip time between the mobile host and the correspondent host, which is approximately 5400 μs in this experimental setup. The flow binding demultiplexing cost varies from 2.3 μs for one binding to 9.2 μs for 60 bindings.

Note that the cost per flow binding decreases as the number of flow bindings in the list increases, with the result converging to about 0.12 μs. This result is explained by the structure of the flow binding lookup function. This function is composed of two parts. The first part takes a fixed amount of time to locate an

Flow Bindings	Packet Processing Time (μs)	Packet Processing Standard Deviation	Demultiplexing by Flow Cost (μs)	Demultiplexing by Flow Per binding (μs)
0	2.1	0.30	N/A	
1	2.3	0.45	0.2	0.20
2	2.7	0.30	0.6	0.30
10	3.9	0.30	1.8	0.18
20	4.7	0.46	2.6	0.13
30	5.3	0.46	3.2	0.11
40	6.7	0.64	4.6	0.12
60	9.2	0.40	7.1	0.12

Table 5

Flow Binding Demultiplexing Cost. The Packet Processing column displays the total time spent by the home agent in deciding how to forward a packet to the mobile host. The Demultiplexing by Flow column displays the cost of flow demultiplexing which is part of the total packet processing time.

entry for the mobile host in the mobility binding table on the home agent by hashing on the mobile host's home IP address. The second part searches into a list for a flow binding corresponding to the incoming packet. When the number of bindings in the list is small, the cost of the first part dominates. On the other hand, when the number of bindings in the list is large, the cost of the first part is amortized, and its effect becomes negligible.

7. Related Work

There are several projects focusing on providing flexibility support for mobile hosts at different layers of the network software stack.

Work at Oregon Graduate Institute and Portland State University [11] concentrates on lower network layers than ours. It addresses the flexibility needs of a mobile host in the face of physical media changes, mainly dealing with IP reconfiguration issues. The focus is on a model that determines the set of available network devices and dynamically reconfigures a mobile system in response to changes in the link-layer environment.

The CMU Monarch project [12] aims at enabling mobile hosts to communicate with each other and with stationary or wired hosts transparently and adaptively, making the most efficient use of the best network connectivity available to the mobile host at any time. It has an overall goal similar to ours, although the project currently does not support the use of multiple packet delivery methods simultaneously. Monarch also focuses on ad hoc networking, which we currently do not support.

The CMU Odyssey project [21] extends the Unix system call interface to support adaptation at the application layer. Odyssey exposes resources to applications by allowing them to specify a bound of tolerance for a resource. When the behavior of the resource moves outside the tolerance window, the application is informed via an up-call.

Making simultaneous use of multiple interfaces is also an important goal of the work by Maltz and Bhagwat [17]. Their approach provides transport layer mobility by splitting each TCP session between a mobile host and a server into two separate TCP connections at the proxy, through which all packets between the mobile host and the server will travel. It allows a mobile host to change its point of attachment to the Internet by rebinding the mobile-proxy connection while keeping the proxy-server connection unchanged. It can control which network interfaces are used for different kinds of data leaving from and arriving at the mobile host. However, their work differs from ours in that it does not use Mobile IP and it selects the interface by explicitly picking the corresponding IP addresses to use on a per session basis, while with our enhancement to Mobile IP, we can select different interfaces to use for different flows even if the mobile host always assumes its static home address.

The BARWAN project [14] at UC Berkeley also provides network layer mechanisms to support mobile hosts, but with a different focus. It aims at building mobile information systems upon heterogeneous wireless overlay networks to support services allowing mobile applications to operate across a wide range of networks. They concentrate on providing low-latency hand-offs among multiple network devices.

The use of a Mobile Policy Table is similar to the Security Policy Database (SPD) in IPsec [15], which needs to be consulted in processing all traffic. While the Mobile Policy Table is used to direct the addressing and routing decisions in sending packets, the SPD deals with the security aspects of traffic control. Another difference is that the Mobile Policy Table is only in effect for outgoing traffic. It affects the incoming traffic through the selection of the source IP address for the outgoing traffic and/or through coordination with the home agent. The SPD is in effect for both inbound and outbound traffic all the time.

In summary, our approach focuses on the network layer, providing increased support for a mobile host to control how it sends and receives packets. Our work differs from other work in the combination of the following features:

1. Mobile IP is an integral part of our system. We address the issue of how Mobile IP can be used most efficiently and flexibly on mobile hosts.
2. Our system provides support for multiple packet delivery methods.
3. Our system enables the use of multiple active interfaces simultaneously.

8. IPv6 Considerations

With IPv6 [10] a mobile host will always be able to obtain a co-located care-of address in the network it is visiting. However, the fact that a mobile host will always have both a home role (acting as a host still connected to its home network) and a local role (acting as a normal host in the visited network) simultaneously will still be an issue. An example is in multicast scoping. A mobile host needs to assume its home role if it wants to join the multicast group scoped within its home domain, while it needs to assume its local role to join the multicast group scoped within the visited domain. Therefore, because of the duality of the roles a mobile host can assume, there is always the need for a mobile host to select different packet delivery methods for different traffic flows.

With regard to multiple interface management under IPv6, we have the following observations:

1. Mobile IPv6 [24] provides route optimization. As a result, the correspondent hosts can send packets directly to the mobile host without going through the home agent. Therefore, the Flow-to-Interface binding extensions need to be sent to the correspondent hosts as well.
2. The IPv6 *flow label* field can be very useful for providing multiple interface support. The flow label is a field of the IPv6 header that is set by the source of the packet and used by routers to identify flows. This flow label field could be used by the Home Agent to demultiplex packets and forward them on the appropriate interfaces without looking into the transport layer field for the port information to identify a flow. This also simplifies the handling of fragmented packets on the home agent by eliminating the need to reassemble them before forwarding them to the mobile host.
3. IPv6 provides a *priority* (class) field used by routers to provide different services to different types of packets. This field can potentially be used in our proposal by the home agent in determining the most appropriate interfaces through which to forward packets addressed to a mobile host in the absence of Flow-to-Interface binding registrations.

9. Conclusions and Future Work

Because of a mobile host's use of multiple interfaces and the dynamic environment in which it may operate, a mobile host using Mobile IP demands flexibility in the way it sends and receives packets. Fine granularity control of network traffic on a per flow basis is needed. Our experiments show that with reasonable overhead we can address the flexibility needs of a mobile host by maintaining a balance between the robustness and efficiency needs of packet delivery.

It is desirable to choose different packet delivery methods according to the characteristics of different traffic flows. We have devised a general-purpose mechanism at the network routing layer to make this possible by coupling the consultation of a Mobile Policy Table with the regular routing table lookup.

It is also desirable for a mobile host to be able to make use of multiple network interfaces simultaneously for different flows of traffic. For this, we have added two new socket options and have extended the Mobile IP protocol with new registration extensions so that a mobile host has more control over which interface to use to send and receive packets.

10. Acknowledgements

We thank Petros Maniatis, Kevin Lai, Mema Roussopoulos, and Diane Tang of MosquitoNet Group for discussions and feedback on the ideas presented in this paper. Ajay Bakre and Charlie Perkins provided thoughtful comments and suggestions on an earlier version of the paper. We are also grateful to the anonymous reviewers for their extremely detailed comments.

This work was supported in part by a student fellowship from Xerox PARC, a Terman Fellowship, a grant from NTT DoCoMo, and a grant from the Keio Research Institute at SFC, Keio University and the Information-technology Promotion Agency, Japan.

References

[1] A. Acharya, A. Bakre, and B. Badrinath. IP Multicast Extensions for Mobile Internetworking. In *Proceedings of the IEEE INFOCOM'96*, 1996.

[2] J. Agosta and T. Russell. *Cellular Digital Packet Data Standards and Technology.* McGraw-Hill, 1996.

[3] M. Baker, X. Zhao, S. Cheshire, and J. Stone. Supporting Mobility in MosquitoNet. In *Proceedings of the Annual USENIX Technical Conference*, 1996.

[4] Computer Emergency Response Team (CERT). IP Spoofing Attacks and Hijacked Terminal Connections. In *CA-95:01*, 1995.

[5] S. Deering and D. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems, Vol. 8, No.2, pp. 85-110*, 1990.

[6] R. Droms. Dynamic Host Configuration Protocol. RFC 2131, March 1997.

[7] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which Employ IP Source Address Spoofing. RFC 2267, 1998.

[8] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2068, 1997.

[9] T. G. Harrison, C. L. Williamson, W. L. Mackrell, and R. B. Bunt. Mobile Multicast (MoM) Protocol: Multicast Support for Mobile Hosts. In *Proceedings of the Third Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1997.

[10] C. Huitema. *IPv6: The New Internet Protocol.* Prentice Hall, 1997.

[11] J. Inouye, J. Binkley, and J. Walpole. Dynamic Network Reconfiguration Support for Mobile Computers. In *Proceedings of the Third Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1997.

[12] D. B. Johnson and D. A. Maltz. Protocols for Adaptive Wireless and Mobile Networking. *IEEE Personal Communications, 3(1):34-42*, 1995.

[13] D. B. Johnson and C. Perkins. Route Optimization in Mobile IP. Internet Draft draft-ietf-mobileip-optim-08.txt (work in progress), 1999.

[14] R. H. Katz and E. A. Brewer. The Case for Wireless Overlay Networks. In *Proceedings of the SPIE Conference on Multimedia and Networking*, 1996.

[15] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. RFC 2401, 1998.

[16] K. Lai, M. Roussopoulos, D. Tang, X. Zhao, and M. Baker. Experiences with a Mobile Testbed. In *Proceedings of the Second International Conference on Worldwide Computing and its Applications (WWCA'98)*, 1998.

[17] D. A. Maltz and P. Bhagwat. MSOCKS: An Architecture for Transport Layer Mobility. In *Proceedings of the IEEE INFOCOM'98*, 1998.

[18] Metricom. The Ricochet Wireless Network Overview. http://www.ricochet.net/ricochet/, 1997.

[19] G. Montenegro. Reverse Tunneling for Mobile IP. RFC 2344, 1998.

[20] G. Montenegro and V. Gupta. Sun's SKIP Firewall Traversal for Mobile IP. RFC 2356, 1998.

[21] B. Noble, M. Satyanarayanan, D. Narayanan, J. E. Tilton, J. Flinn, and K. Walker. Agile Application-Aware Adaptation for Mobility. In *Proceedings of the 16th ACM Symposium on Operating System Principles*, 1997.

[22] C. Perkins. IP Encapsulation within IP. RFC 2003, 1996.

[23] C. Perkins. IP Mobility Support. RFC 2002, 1996.

[24] C. E. Perkins and D. B. Johnson. Mobility Support in IPv6. In *Proceedings of the Second Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1996.

[25] J. Postel. Internet Protocol. RFC 791, 1981.

[26] DARPA Internet Program. Transmission Control Protocol. RFC 793, 1981.

[27] S. H. Redl, M. K. Weber, and M. W. Oliphant. *An Introduction to GSM*. Artech House, 1995.

[28] Lucent Technologies. WaveLAN Support. http://www.wavelan.com/support/index.html, 1998.