IP-based Protocols for Mobile Internetworking*

John Ioannidis
Dan Duchamp
Gerald Q. Maguire Jr.

Department of Computer Science
Columbia University
New York, NY 10027

ji@cs.columbia.edu

CUCS-006-91

Abstract

We consider the problem of providing network access to hosts whose physical location changes with time. Such hosts cannot depend on traditional forms of network connectivity and routing because their location, and hence the route to reach them, cannot be deduced from their IP address. We present protocols that seamlessly integrate mobile hosts into the current IP networking infrastructure. They are primarily targeted at supporting a campus environment with mobile computers, but also extend gracefully to accomodate hosts moving between different networks. The key feature is the dependence on ancillary machines to track the location of the mobile hosts. Our protocols are designed to react quickly to changing topologies, to scale well, and not to place an overwhelming burden on the network.

1 Motivation

In recent years we have observed a proliferation of portable computers. There is a strong trend toward the production of smaller and more powerful such units. A serious drawback of current portable computers is that their users have no access to their everyday work environment from their portable; they have to use awkward file transfer mechanisms to upload and download any files they need while away from their office. There is a strong desire, especially among computer-literate users, to have the same environment and access to the same data both from the workplace and far from it. While modern portable computers are capable of providing this functionality, they are hindered by the lack of continuous network connectivity.

The logical step is the addition of highly available, high-bandwidth mobile networking capabilities to portable computers. The infrastructure to support this concept is slowly and haphazardly

*Accepted for publication at SIGCOMM'91.

appearing. Beyond the near-ubiquitous LAN in business environments, high-speed wireless interfaces are starting to appear at speeds of up to 2Mbps, using radio frequency (RF) or infrared (IR) technology. Moreover, there has been a significant increase in the availability of alternate long-haul networking resources, such as dial-up IP services, companies and universities offering SLIP connections to their employees and students, and so on.

In this paper, we present a set of protocols that enable mobile computers to move around the Internet and retain their connections and addressability. These mobile computers may be equipped with either wired networking interfaces (Ethernet, Token-Ring, SLIP), or wireless ones (RF, IR). In both cases, a combination of software on the portables themselves as well as on infrastructure machines ensures continuous addressability. Our protocols work within the Internet suite of protocols, and provide mobile hosts with IP-level connectivity. They are designed to impose a minimal burden on the underlying networks by minimizing the amount of extra traffic needed, scale well, and respond quickly to changes in topology. Furthermore, no changes are needed to the software of non-participating hosts, or that of gateways.

2 Protocols for Mobile Internetworking

2.1 The Model

In this paper, the term 'mobile' implies 'able to move *while retaining its network connections*'. Thus, we shall call a host that can move while retaining its network connections a **mobile host**, or MH for short. The infrastructure machines that support the MHs we shall call **Mobile Support Stations**, MSS for short. A logical or geographical segment of the campus supported by an MSS is called a **cell**. Our protocols distinguish between 'local' and 'remote' networks. This is a result of the standard partitioning of the 32-bit IP address into a "network" part and a "host-within-network" part. We call host mobility within the realm of one network number (i.e., within a single campus) **intracampus** mobility, and mobility between networks (i.e., between different campuses) **intercampus** mobility. The functionality needed for the two cases is largely the same. We shall consider that the MHs are on their own subnet(s), or that they occupy a contiguous set of addresses in the address space (which is a more accurate description if the campus network is not subnetted). This facilitates intra-campus routing and immediate recognition of a host as an MH. If the campus IP address space is heavily populated, it is acceptable that the MHs have their own network number.

The MH always retains one IP address, called its **home address**, regardless of where it is on the network. The mobile host relies on ancillary machines (the MSSs) to maintain its addressability. Failure to maintain a home address would be both inconvenient and inefficient – higher-level software may have to be modified to deal with changing addresses; network users often associate other users with particular machine names; nameservers may not respond quickly enough to changes in name/address mappings, etc.

Our design philosophy is that the burden of providing mobile internetworking should fall mainly on its users, and that hosts not using these features should continue running with their ordinary software. Thus, although it may be desirable in order to enhance performance, it is not necessary to run special routing code on gateways and bridges.

Let us now consider the infrastructure necessary to support *intra*campus mobility. All the hosts

Figure 1: Prototypical campus environment

2.2 Overview of the Protocols

Given the above prototypical campus, let us see how communication takes place. Suppose a wandering MH, `m5`, moves into `s2`'s cell. The MSS of the cell periodically broadcasts a *beacon*, so `m5` receives `s2`'s beacon, realizes that it is in `c2`, sets `s2` as its default gateway, and identifies itself. If it were previously in another cell, say `c4`, it will also tell `s2` that its previous MSS was `s4`. `s2` will then notify `s4` of `m5`'s migration. `s2` must acknowledge `m5`'s greeting. `m5` will continue trying to identify itself to `s2` until it receives the acknowledgement. `s2` will also expect an acknowledgment for the forwarding pointer it sent to `s4`. It may not get this acknowledgement immediately because

[1] A 'connected network' is one where link-level broadcasts such as ARP requests are received by all machines.

of a network partition, so the MSS should function correctly even before receiving the forwarding acknowledgement. Retransmissions should follow a bounded binary or linear exponential backoff so as not to adversely affect the network in the case of such a partition.

MHs are always configured to have the MSS of the cell they are in act as their default gateway. Other hosts are configured to gateway traffic destined for the MH subnet (or the range of addresses) through the MSS on their connected network. If no MSS exists on a particular segment, the default routes through that segment's gateway will eventually lead to some MSS.

There are four distinct scenarios involving MHs. First, an MH may send a datagram to another MH in the same cell. Next, an MH may send a datagram to a 'regular' host. Further, a regular host may send a datagram to an MH. Finally, an MH may send a datagram to an MH in a different cell. To illustrate these four cases, consider the following examples. We shall refer to Figure 1 again.

1. When `m1` sends a datagram to `m2` (an MH in the same cell), the normal address resolution mechanism (e.g., ARP) will deliver the datagram to its destination. The MSS is not involved.

2. When an MH sends a datagram to a regular host, (e.g., `m2` to `h3`), it automatically gateways through its MSS. Normal IP routing will deliver the packet to its destination.

3. When a regular host, `h3`, wants to talk to an MH, `m3`, its route for the MH subnet, `n9`, will be through the local MSS, `s2`[2]. The first time this happens, `s2` will query the other MSSs in order to locate the one responsible for the `m3`. This query may be multicast, sent individually to all MSSs, or flooded from MSS to MSS. `s1` will respond to this query, and `s2` will cache the information. Then `s2` will encapsulate the datagram in a special packet and send it directly to `s1`. Upon receipt, the latter will unpack the original datagram and deliver it to `m3`.

4. When an MH sends a datagram to another MH which is not in its cell (e.g., `m4` to `m6`), the MSS, `s2`, will 'proxy-ARP' for `m6`, discover `s3` as in the previous case, and deliver the datagram.

We call the protocol that does the encapsulation **IPIP** ("IP-within-IP"), and the protocol used to exchange information among MSSs, **MICP** ("Mobile Internetworking Control Protocol"). They are both described in detail in the following sections.

There is an interesting special case to consider: an island of a wired network, `n2` may exist, which is attached to the rest of the network by a pair of machines with wireless interfaces, `s5` and `s1`. In this case, routing table entries are set up as expected, and packets for `n2` are eventually sent to `s1`. The latter knows that the route for `n9` is through `s5`.

*Inter*campus mobility is assumed to be unusual, and is handled as a variation of the intracampus case. An MSS is needed on any segment of the remote campus that wants to support intercampus mobility. Some MSSs on the home campus should be able to collaborate with remote MSSs: the campus gateway(s) should have a route to one or more MSSs so that incoming packets destined for MHs can be routed properly. Referring once again to Figure 1, there are no MSSs directly connected to `n0`; if the campus MSS is `s4`, the route for subnet `n9` on the campus gateway will be `gw3`, and the route to `n9` on `gw3` will be `s4`.

[2] There is also the case where a host is on a segment with no MSSs; in that case, a string of gateways will eventually deliver the datagram to a segment *with* an MSS, and then the process will continue.

Upon entering a cell in the foreign campus, the MH will realize that it is in a different network. It will then ask the local MSS for transient IP address which it will use for addressing. Now, all packets leaving the MH will be encapsulated in an IPIP packet addressed either to the local MSS, or to the MSS in the MH's home campus that handles such traffic. The address of the home MSS can be specified with a configuration command, or it can be a well-known address in the MH's home network. The MH will then send a greet packet to its home MSS which will notice that the packet comes from a foreign network, but claims to be an MH for its campus, It will deduce that this is an MH in a foreign campus that needs tunneling service. From then on, this MSS will handle traffic for the MH.

2.3 Hardware Requirements

Nothing is *required* from the hardware that is not already commercially available. In the case of wireless mobile hosts, the ability to detect signal strength and report it to the device driver will be helpful in realizing that the host is at the outer reaches of a radio cell and that it should try to switch to a different one. Also, in the case of spread-spectrum or multi-channel radio communications, the ability to receive on multiple 'channels' at once is highly desirable (so as to detect areas where radio cells overlap and when the signal from one is stronger switch from one MSS to another).

2.4 The Data Link Layer

Two interactions have to take place in the data-link layer, before IP addressability has been established. These are the **beacon** protocol that the MSS uses to advertise itself, and the **dynamic IP address assignment** protocol that the MH and the MSS use to negotiate a transient IP address for the MH in the case of intercampus mobility.

2.4.1 MSS Beacon Protocol

MHs entering a new cell must know how to contact the MSS in order to establish themselves. The two alternatives for such an identification protocol are (a) have the MSS periodically broadcast its identity, and (b) have the newcomer MH broadcast a request for the local MSS. The former is more appropriate because this way only one packet (rather than a broadcast/response pair) is required, and because MHs already in the cell can tell that they are still within the realm of the MSS. On hardware with signal strength detection, the strength of the beacon may be used to determine whether it is time to seek another MSS. Last, transmitting uses more power than receiving, and since the portable MH might be battery-powered, it is desirable to conserve power. The last two issues apply only in the case of wireless communication. Another advantage is that the MH does not have to reveal its presence unless it needs to communicate. Figure 2 describes the packet periodically broadcast by each MSS.

The only flag currently defined is whether this cell supports dynamic IP address assignment, in which case this field is 1. The next doubleword is the IP address (in network order) of the MSS on this side of the cell, and the third doubleword is the netmask of this cell. The latter is needed in order to set up the IP broadcast address and the default routes properly. Upon receipt of this

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

BEACON	flags	checksum
IP address of MSS		
netmask of this cell		

Figure 2: Beacon protocol

packet, the MH should set its default gateway and netmask to the values supplied in the packet, and send the MICP_GREET packet (described later) to the MSS.

An interesting case is what happens when the MH receives beacons from two MSSs at a time. In the case of wireless cells, if the hardware gives an indication of the signal strength, the MH should stay with the cell that has the stronger signal, Otherwise, the MH may wait until it no longer receives the beacon from the previous MSS before it switches to the new one. In the case of wired cells, it may stay with the MSS whose beacon it hears first. A more complex solution may also be adopted, e.g., involving information about the load in the cell. The exact behavior in the presence of multiple beacons has yet to be investigated.

2.4.2 Dynamic IP-address Assignment Protocol

When an MH moves into a foreign campus, it requests a transient IP address which it will use to talk to the local MSS, and also to encapsulate IPIP packets in order to maintain open connections to its home campus. The MH knows it moved into a foreign campus if the network portion of the MSS's IP address is different from its own. The request for a dynamic address is shown in Figure 3.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

IPADRQ	version	checksum
home IP address of MH		
default MSS in home network (if any)		

Figure 3: Transient Address Request packet

This packet may be broadcast to the local link-level broadcast address, or (if known from a beacon packet), to the link-level unicast address of the MSS. Upon receipt, the MSS selects an unused IP address from that network and gives it to the requesting MH in the form shown in Figure 4. It is easy to devise algorithms for selecting such transient IP addresses; we will not address them in this paper.

The MH periodically (every "timeout-value" seconds, as specified in the IPADGR packet) sends a packet confirming that it is still in the cell, but only if there has been no other traffic between those two machines in that period. The proper value for the timeout packets will have to be determined experimentally. Arguably, this functionality can be moved to a higher-level protocol, but since it interacts with timeout tables kept in the data-link driver, it may as well be here. The format of the packet is shown in Figure 5.

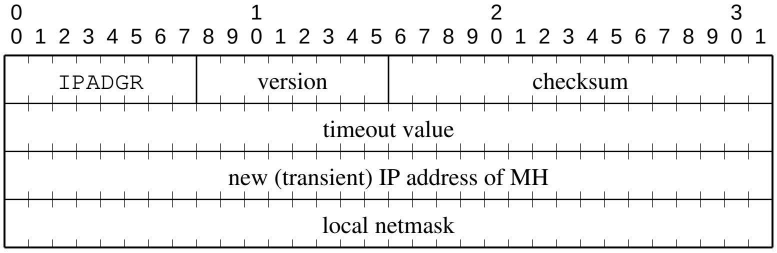

Figure 4: Transient Address Grant packet

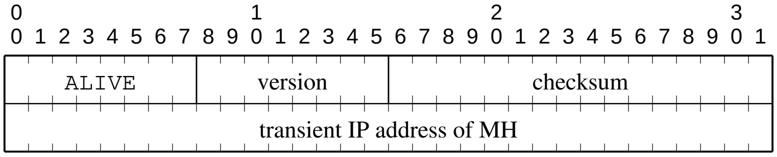

Figure 5: MH Still Alive packet

2.5 The Network Layer

In Section 2.2 we sketched how datagrams are sent from and delivered to MHs. The IP layer must be modified to decide how to route a datagram, and when to use IPIP and MICP. Both protocols use IP to receive and deliver their control packets, so conceptually they sit on top of IP, but they control the operation of IP, so in fact they sit right next to IP. This is analogous to the way ICMP works. The The following two sections describe those protocols.

2.5.1 The IP-within-IP protocol

The IPIP protocol is used to 'tunnel' IP datagrams from one cell to another, or from one network to another, essentially using IP as a data-link protocol. In a sense, it is equivalent to IP loose source routing [Pos81b]. When an IP datagram is received by the IP layer, it is first reassembled (if it was fragmented), and then the routing tables are consulted. If the destination is in another cell (with a known MSS), it is encapsulated in an IP datagram of type `IPPROTO_IPIP`, and sent to the remote MSS. Thus, the IP destination and source addresses of the datagram are the two endpoints of the tunnel. Upon receipt by the remote MSS, the IPIP protocol module strips the IPIP header and feeds the packet (which now has the real destination and source IP addresses) back into the IP queue so that it can be delivered. The packet format is shown in Figure 6.

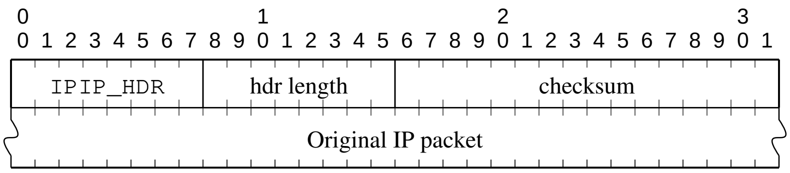

Figure 6: IPIP packet format

`IPIP_HDR` is a constant indicating that the datagram is encapsulating another IP datagram. *Hdr length* is the size of the header in octets.

The benefit of our approach is that it does not rely on elaborate routing schemes, per-host routing on gateways, or on possibly poorly implemented options (loose source routing), and the amount of overhead involved is similar to other tunneling methods (see, e.g., [WPD88]). IPIP can also be used as a general-purpose protocol whenever IP-level encapsulation is needed.

2.5.2 The Mobile Internetworking Control Protocol

The Mobile Internetworking Control Protocol is used by the MSSs to exchange control information, and by the MHs to signal to their respective MSSs that they have changed cells. MICP datagrams are encapsulated in IP datagrams with protocol type `IPPROTO_MICP`. The following MICP functions are defined, along with their packets:

MH Greeting:

This is the packet that the MH sends to the MSS in response to a beacon. The format is shown in Figure 7.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

MICP_GREET	0	checksum
IP address of MH		
IP address of previous MSS (or 0)		
timestamp		

Figure 7: Greeting Packet

Upon receipt of this packet, the MSS will update its tables to show that this particular MH is now in its cell, will acknowledge the greeting, and will also send a forwarding packet to the previous MSS the MH was talking to (if any). It will continue sending this packet until acknowledged. The *timestamp* is the MH's concept of real time; it usually is the number of seconds since some fixed date. Greeting packets are idempotent; if the MSS knows that is handling a certain MH, it will reply to the greet, but will otherwise ignore it.

Greeting Acknowledgement:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

MICP_GRACK	0	checksum

Figure 8: Greeting Acknowledgement

The packet shown in Figure 8 is sent to the MH after the MSS receives the greeting. If the MH does not receive this acknowledgement within a certain amount of time, it will resend an `MICP_GREET` packet. This handshake is needed is because the MH must be certain that the MSS knows about it (otherwise traffic destined for the MH will not be able to reach it). If the greeting packet is lost, then the MH will simply resend it. If the acknowledgement packet is lost, the MH will send another greeting until it gets an acknowledgement.

Forwarding Pointer:

MICP_FWDPTR	0	checksum
IP address of MH		

Figure 9: Forwarding Pointer Packet

When an MSS receives this packet (shown in Figure 9), it records the forwarding address of the MH. This forwarding address is the source IP address of the packet. It is cached for some time, and then flushed. This timeout needs to be determined experimentally. The reason for this caching is that other MSSs cache the addresses of the MSS that serves a particular MH, and may try contacting this MSS.

Forwarding Acknowledge:

MICP_FWDACK	0	checksum
IP address of MH		

Figure 10: Forwarding Pointer Acknowledge

An MSS must acknowledge a forwarding pointer in order to make certain that the previous MSS will not attempt to handle traffic for an MH that it no longer serves. Figure 10 shows the format of this packet.

Redirecting:

When an MSS receives a datagram for an MH that it no longer serves, but for which it has in its cache the address of the MSS currently serving it, it will send a redirect to the sender. To avoid having to rely on timeouts from higher-level protocols, the MSS will also attempt to deliver the received datagram to where it should have gone. Figure 11 shows the format of the redirection packet.

MICP_REDIR	0	checksum
IP address of MH		
IP address of new MSS		

Figure 11: Redirect Packet

Assuming that the MH does not change cells faster than these packets can travel, the network should be able to track the MH at all times. The only problem with this redirect packet being lost is that the originating MSS will not be informed of the MH's movement. This does not cause problems; if another packet is received, another redirect will be sent. Eventually, one will reach the offending MSS. Hence, no explicit acknowledgement to the redirect packet is needed.

Expired MSS Information:

If an MSS receives an IPIP datagram for an MH that it does not handle, and for which it has no information in its cache, it returns the undeliverable IP packet to the sender. The sender will subsequently go through the peer MSS discovery process described next. The format of this packet is described in Figure 12. The loss of this packet will result in a similar scenario as the loss of a redirect; thus, no acknowledgement of this packet is needed.

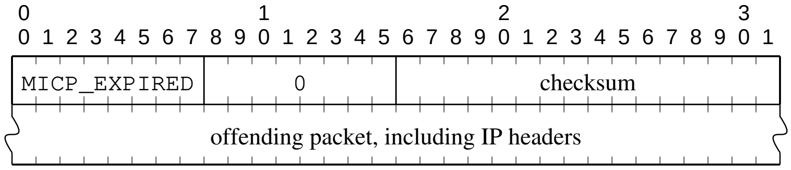

Figure 12: Expired Forwarding Entry packet

Peer MSS Discovery:

When an MSS is asked to deliver a packet to an MH on another segment, it has to discover which MSS is responsible for that particular MH. It queues the packet and then *multicasts* [Dee89] a message to all the MSSs on the campus, asking them to respond if they are responsible for that MH. If the appropriate MSS is found, the packet is sent to the peer MSS using IPIP. An MSS may ask for more than one MH at a time.

If multicasting is not supported on that campus or on the MSS, the MSS may simply unicast such a request to all the MSSs on campus. (We describe how to discover all MSSs on campus later). Alternatively, each MSS may keep a list of 'neighboring' MSSs, and propagate a WHOHAS request to all its neighbors (except the one it received the request from), until the MSS that actually handles the MH is reached. This is essentially flooding. The TTL value in the header is decremented every time the packet is propagated. This 'on-demand' discovery of peer MSSs allows them to know only of the MSSs involved in their MHs' traffic. The alternative would be to have each MSS notify all the others every time an MH moved into its cell, which would result in unacceptably high traffic. Figure 13 shows the format of the WHOHAS request.

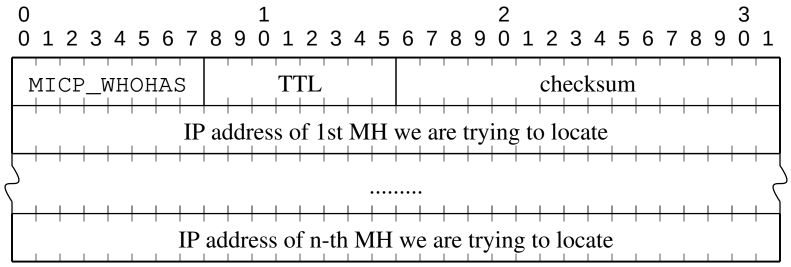

Figure 13: MH Discovery Packet

When an MSS receives such a packet, it searches its tables to see if it is indeed the MSS for the requested MH. If so, it replies with a packet of the form shown in Figure 14. Its return IP address can be recovered from the IP header, and therefore it is not necessary to include it in the packet.

Depending on the size of the network and the traffic, it may be advantageous for all MSSs to periodically broadcast their MH tables. The conditions under which this works better are to be determined experimentally and by simulation. The timestamps are necessary to ensure correctness in case more than one reply is received for the same MH. This can happen when, e.g., an MH has

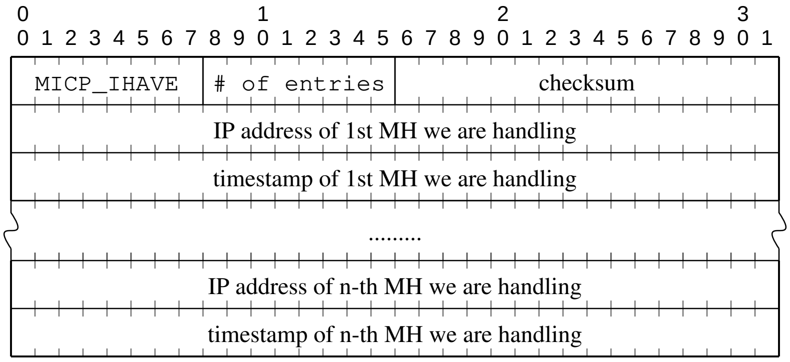

Figure 14: MHs Handled by an MSS

just changed cells; the new MSS has received the greeting, but the old MSS has not yet received the forwarding pointer. Then they will both reply to an MICP_WHOHAS request; however, only the newer MSS's reply will be considered since the timestamp will be newer. Note that the timestamp from an MH is only compared against previous timestamps from itself. At no point is the timestamp compared to a reference time base. All that is required of the timestamp is that it increase with time, and that the granularity is large enough that no wrap-arounds occur while the MH is in use. A 32-bit timeout counting seconds is thus sufficient.

If either the MICP_WHOHAS or the MICP_IHAVE packet is lost, the procedure will repeat when the MSS retries to send the IP packet that caused the lookup. If the MSS does not receive any MICP_IHAVE replies after a certain number of retries, it gives up and sends an ICMP_UNREACH_HOST message [Pos81a] to the machine trying to reach the MH in question.

Discovery of All MSSs On-Campus:

As discussed above, in order for an MSS to deliver an IPIP datagram to another segment of the network, it must know the address of the MSS handling that particular segment or cell. The cleanest way is for all MSSs on a campus to support multicast addressing and be members of the same "host group," and for the MICP_WHOHAS packet to be sent to that "multicast host group." However, this requires the additional support for multicasting, something that is not yet widespread.

If the entire campus is on the same physical or logical LAN (no selective repeaters, bridges or gateways), then the obvious solution is to broadcast the request. If this cannot be done, then something more complex is needed. We cannot assume any 'intelligence' from the selective repeaters or gateways, so we shall let the burden of efficient communication between MSSs rest on the MSSs themselves. Two of the ways we suggest are:

Each MSS knows the IP addresses of all the other MSSs on campus, and it queries all of them every time it needs them. The list of MSSs can be set up statically at configuration time. Alternatively, every time a new MSS is added, it is given the address of one of the others. At that point, it sends a 'I am a new MSS' packet to that machine, and the recipient replies with *its* list of MSSs. The new MSS then proceeds to inform all the others on that list that it is a new machine. Figures 15 and 16 show the packet formats.

The SEND/DONTSEND subcode in the first packet controls whether the recipient MSS will reply with its list of MSSs. The address of the requesting MSS can be extracted from the IP header.

0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

MICP_NEWMSS	subcode	checksum

Figure 15: New MSS request/announcement

0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

MICP_MSSLST	0	checksum
IP address of first MSS on the list		
....		
IP address of last MSS on the list		

Figure 16: New MSS request/announcement

Alternately, each MSS only knows its immediate neighbors. It registers itself with a MICP_NEWMSS with subcode NEIGHBOR. Now, every time an MICP_WHOHAS packet is sent, it is sent to the neighboring MSSs. If they have the MH, they reply, otherwise they propagate the request to their other neighbors (if any). This problem is analogous to the discovery of neighbors in an interior-gateway protocol. If we can run our protocols on the campus or autonomous systems gateways, then the MSS discovery can happen in an self-organizing fashion. Otherwise, at least some of the work has to be done manually.

2.6 Higher-Level protocols

The existent transport layer (and up) protocols need no modification. The whole purpose of our protocols is to provide transparent network access to anything above the network layer. There may be some minor problems, though. Some implementations of TCP use 'keep-alive' packets on idle connection to probe the remote end and verify it is still alive. This may adversely effect TCP connections on an MH that stays without an MSS for longer periods of time than the keepalive mechanism would allow. Such timeouts are usually on the order of a few minutes, so cell switches will not affect them, but prolonged absences might. Thus, the use of keep-alives is discouraged in the presence of mobile networking.

Protocols described thus far attempt to hide mobility; however, it may be the case that high-level software can benefit from knowledge about relocation. For example, access to system programs and libraries does not have to go through the entire IPIP process if there are copies of those resources available locally. There has been a lot of research and development on access of replicated resources (see, e.g., [Gif79]). It should be fairly easy to adapt most of these schemes to work in our idea of a mobile environment.

3 Evaluation

Implementation of the protocols is still underway; no performance figures are available yet. In this section we shall give a qualitative analysis of the complexity of the structures and algorithms involved, and the expected performance of our protocols.

3.1 Performance Issues

Cell Switch Latency: The time needed to integrate an MH into a cell is determined by how often the MSS sends its beacon. Once the MH receives the beacon, it will respond with an `MICP_GREET` packet. If the MSS transmits its beacon once a second, the operation should take under a second on the average. The beacon packet itself is 12 octets. In an ethernet-like medium, there will be another 14 octets of header information, for a total of 26 octets or 208 bits. Even on a slow radio link running at 250kbps, and allowing for frame preambles and postambles, sending once a second uses less than 0.1% of the bandwidth.

MSS **Load:** An MSS should be fast enough to process all of its cell's traffic so that it can proxy-ARP, gateway packets destined for machines outside the cell (optionally adding/stripping IPIP headers), and also keep up with MICP traffic from other MSSs. The MSS can take advantage of the fact that the MH addresses are contiguous, and use the 'host' part of their address as an index to an array when looking up information about that MH. Hence, with a single memory lookup, the ARP code can decide whether it should send a proxy-ARP reply to an MH or a host wishing to talk to an MH in a different cell, and the tunnelling code can find the required IPIP information.

IPIP Overhead: Encapsulating IP datagrams in an IPIP packet involves prepending them with a new IP and IPIP header. Since this header only changes when the corresponding MH moves, the system can store the entire header and keep a pointer to it[3]. The overhead of IPIP is 24 octets (20 for the new IP header and 4 for the IPIP header). For small IP packets (e.g., from `rlogin` or `telnet` sessions), this may be a significant fraction of the total packet size. The data rate for such traffic is low (since it is limited by the user's typing speed), and hence this should not pose a problem. In the case of large packets (from file transfers, mail delivery, etc.) the added IPIP header is negligible; however, long packets may cause an unfragmented IP packet to fragment. Further research is needed on how frequent this problem will be, and if it proves to be a problem, how to dynamically modify the Path MTU (in a manner analogous to [MD90]) in order to limit it.

MICP Time and Space Complexity: There are two kinds of MICP traffic: MSS configuration traffic and MH-related MICP messages. The first kind creates negligible traffic; MSSs change location very rarely. When an MH moves to a new cell, it sends one packet (the greeting) to its MSS. The MSS acknowledges this packet, it sends a forwarding packet to the MSS previously handling the MH, and expects a reply. The `MICP_WHOHAS/MICP_IHAVE` handshake takes place only the first time an MSS has to locate an MH. After that, the MSS caches the MH's location[4]. If the MH moves, the MSS that was previously responsible for it will send the `MICP_REDIRECT` as well as deliver the packet. Cached entries expire if they have been idle for longer than a specified period of time, and if communication with that MH is required again, then the `WHOHAS/IHAVE` process is

[3] On systems using BSD-style `mbufs` or System-V STREAMS, prepending a header to a packet involves moving pointers; no copying has to take place.

[4] The format of that information is implementation-dependent. For example, it could be the entire IPIP header.

repeated.

Thus, the cost of initially locating an MH in terms of packets is proportional to the number of MSSs on campus, and the cost of moving is four packets plus one packet per cell switch per connection. Also, each MSS has to know only the locations of MHs that machines in its cell(s) are communicating with, and this should be much smaller than the total number of MHs on a campus. The communication patterns are not expected to be random; rather, a great deal of both spatial and temporal coherence is expected, and that should further reduce the caching requirements.

Fault Tolerance and Load Sharing: In areas with a lot of MH traffic, more than one MSS may be set up. Since an MH usually responds to the first beacon it receives, the MHs will be distributed evenly among the MSSs present. More elaborate load-sharing and MSS-selection schemes may also be considered. When an MSS goes down, the MHs it served will stop receiving its beacon; after a short timeout period (to be determined), they will try to locate another beacon and they will find one of the other MSSs. This is as if the MH had moved into another cell.

Scalability: A natural concern is how well our approach scales as the number of MHs increases. We assume that there are enough MSSs to handle all the MHs, and that the campus (wired) network would have been able to handle the load were the MHs to be connected as regular hosts on the network. As we have seen the MH-to-host traffic involves only a small amount of IPIP overhead. The real problem is the duplicated traffic when a regular host sends to an MH supported by an MSS which is not on the same connected network. In this case, the MSS on that connected network will receive the datagram, encapsulate it in IPIP, and resend it to the remote MSS. Thus, the amount of data sent on the connected network doubles for these cases. Once this traffic becomes a considerable fraction of the network bandwidth, it will be desirable to have the segment gateway also handle MH-related traffic. To conclude, scalability concerns should be compared to the alternative of just reconfiguring the addresses of MHs every time they change cells (in which case they would appear as regular hosts), not to an MH-less campus; our approach fares well in this comparison.

Higher-level Software: A mechanism for notifying applications that the host has moved should be considered. Applications may use this information to decide when to switch from one replicated resource to another, etc. In a UNIX environment, e.g., processes may register with the MICP layer to receive a signal or out-of-band data whenever the MH changes cells.

3.2 Discussion of Alternatives

In trying to decide how to structure our protocols, we considered various alternatives. Once again, our primary objective was host addressability throughout the network, and the design philosophy was to do so in a scaleable and efficient way, in the framework of today's Internet.

Another consideration is that not all MSSs need to know the whereabouts of every MH on campus. Only the machines involved in a connection should be informed. Therefore, we do not need elaborate schemes for informing the entire network of the location of an MH. Rather, the discovery of MHs is a demand-driven process. The overhead of this discovery is low enough that we do not need to worry about delays.

Given unlimited networking resources (or a very small network), the obvious solution is to have each host register with a local gateway, and have gateways keep track of the location of all the other hosts in the internet. Clearly, such a solution does not scale well, and the amount of traffic necessary to follow each host is prohibitive.

Another approach mobility would be to acquire a new address in each cell, and then notify all higher-level software of the change. As it has already been explained, this would be highly undesirable and inefficient. The decision to keep one address protects higher-level software, and the overhead involved in maintaining the MSS structure is ostensibly lower than updating name servers.

4 Related Work

As early as 1980, [SP80] described addressing of mobile hosts in the Arpanet environment. The approach was also that mobile hosts should be on their own virtual network, and a global (possibly distributed) database of mobile hosts in conjunction with source routing would take care of tracking moving hosts. This scheme does not scale well as the number of mobile hosts increases. Our scheme maintains very little state for each mobile host, and uses caching and timeouts to keep the size of the location tables low.

A form of tunneling is described in [WPD88]. The object is to use the Internet as a virtual network to deliver an IP datagram with a multicast (class-D) destination address between two networks separated by routers that do not know about multicast routing. The approach (called 'weak' encapsulation) is to put a loose source route in the packet, with the originating host and the multicast destination as its two elements, and use the two endpoints of the 'tunnel' as the IP source and destination address. Intermediate gateways (that do not know about class-D networks) will ignore the loose source (since it is pointing to a class-D address, and they do not know how to deliver), and eventually deliver the datagram to the receiving end of the tunnel. The receiving end knows about this tunnel mechanism, detects the fact that this is a tunneled packet, restores the source and destination addresses from the loose-source-route option, and delivers the multicast datagram to its intended recipient(s). In the case of multicast routing, this also preserves the benefits of the other IP options, the time-to-live, etc. In our case, this will not be a good mechanism because:

- the destination address is a valid (unicast) IP address, so some other mechanism must be used for fooling the intermediate gateways into forwarding this packet until it reaches its destination.
- there is still a chance that intermediate gateways may not process or even pass, source-routing options correctly (there are still gateways that do not correctly implement all IP options).

Cellular telephony deals with mobile units that can 'roam' (move from cell to cell) while maintaining their connection. However, the mode of operation is different, as there is one virtual circuit per phone, control is much more centralized, and the underlying network is different.

Fowler has investigated the problem of general object movement in distributed systems. The paper abstracted from his dissertation [Fow86] gives a detailed analytical performance analysis of three different forwarding strategies. The strategies are for scenarios in which forwarding pointers form a tree; i.e., possibly many locations forward to the current location, which is the root of the tree. Fowler's work is a complexity analysis of the following three protocols, with the unsurprising result that **pcacc** is superior:

lacc: A chain of pointers from the starting location to the current location. The chain is never compressed and so access to the object forever uses this chain.
jacc: A site in the tree, upon next access, updates its forwarding pointer to point to the root.
pcacc: A site in the tree, upon next access, updates the forwarding pointers of all other sites in the tree, thus collapsing the tree to one level only.

Awerbuch and Peleg [AP89] discuss the use of a combination of forwarding pointers and router updates, with the cases of short-distance and long-distance moves used to determine the level of effort invested in updating routers: nearby routers are updated, faraway routers are not – messages from faraway sites must be forwarded. Although this work tackles few practical problems, it is valuable in that it provides an intelligent answer to the question of where to "draw the line" between re-mapping lookup tables and not doing so.

5 Summary

We have presented an infrastructure that enables mobile machines to keep their network connections while they move in a networked environment. We have distinguished intracampus from intercampus movement, and have focused on the intracampus case as we believe that this is the more interesting and useful case, and because intercampus is based on the existence of intracampus mobility. The two protocols we have defined, IPIP and MICP, work with IP to provide continuous network connectivity without affecting higher-level protocols and software. While we do not have any performance measurements yet, our qualitative analysis shows that our protocols will respond quickly to changing topologies, and that very little state and communication is necessary to track the mobile machines as they change location.

6 Acknowledgments

This work was supported in part by IBM Corporation. We thank Steve Deering (Xerox PARC) and Phil Karn (Bellcore) for providing feedback and historical information.

References

[AP89] B. Awerbuch and D. Peleg. Online tracking of mobile users. Technical Report TM 410, MIT LCS, August 1989.

[Dee89] S. Deering. Host extensions for IP multicasting. RFC 1112, August 1989.

[Fow86] R. J. Fowler. The complexity of using forwarding addresses for decentralized object finding. In *Proc. Fifth Ann. Symp. on Principles of Distributed Computing*, pages 108–120. ACM, August 1986.

[Gif79] D. K. Gifford. Weighted voting for replicated data. In *Proc. of the Seventh Symp. on Operating System Principles*, pages 150–162. ACM, December 1979.

[MD90] J. Mogul and S. Deering. Path MTU discovery. RFC 1191, November 1990.

[Pos81a] Jon Postel. Internet Control Message Protocol. RFC 792, September 1981.

[Pos81b] Jon Postel. Internet Protocol. RFC 791, September 1981.

[SP80] C. Sunshine and J. Postel. Addressing mobile hosts in the ARPA Internet environment. IEN 135, March 1980.

[WPD88] D. Waitzman, C. Partridge, and S. Deering. Distance-Vector multicast routing protocol. RFC 1075, November 1988.