IP-based Protocols for Mobile Internetworking

John Ioannidis, Dan Duchamp, Gerald Q. Maguire Jr.
ji@cs.columbia.edu, djd@cs.columbia.edu, maguire@cs.columbia.edu
Department of Computer Science
Columbia University

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 network address. In this paper, we explore the concept of providing continuous network access to mobile computers, and present a set of IP-based protocols that achieve that goal. They are primarily targeted at supporting a campus environment with mobile computers, but also extend gracefully to accommodate hosts moving between different networks. The key feature is the dependence on ancillary machines, the Mobile Support Stations (MSSs), to track the location of the Mobile Hosts. Using a combination of caching, forwarding pointers, and timeouts, a minimal amount of state is kept in each MSS. The state information is kept in a distributed fashion; the system scales well, reacts quickly to changing topologies, and does not place an overwhelming burden on the network.

1 Motivation

In recent years we have observed a proliferation of portable computers, and a 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; the users have to use awkward file transfer mechanisms to upload and download any files they may need while away from their office. There is a strong desire, not only among computer-literate users, to have the same environment and access to the same data both in the workplace and away 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 technology to support this concept is appearing. Beyond the near-ubiquitous LAN in business environments, high-speed wireless interfaces are starting to appear at speeds of up to 16Mbps, using radio frequency (RF) or infrared (IR) technology [Dav91]. 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. All these activities suggest, among other things, a user base eager to have high-bandwidth access to computation and information resources.

In this paper, we explore the concept of providing continuous network access to mobile computers, and present a set of IP-based protocols that achieve that goal. 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. Protocols above the network (IP) layer are shielded from this mobility; protocols at the transport layer and above need no modification; they continue to operate as if they were on ordinary, static hosts. Finally, no changes are needed to the software of non-participating hosts or gateways.

2 Mobile Internetworking Protocols

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 Sta-**

tions, 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 or organization) **intracampus** mobility, and mobility between networks (i.e., between different campuses) **intercampus** mobility. The functionality needed for the two cases is largely the same.

The MH always retains one IP address, called its **home address**, regardless of where it is on the network. Ancillary machines (the MSSs) assure its addressability. Failure to maintain such a single 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; name servers may not respond quickly enough to changes in name/address mappings, etc. The MH may have more than one network interface; in this case, it may use as its home address the address of one of its interfaces, or use a totally unrelated address. Furthermore, it may be convenient to consider that the home addresses of MHs are on separate subnet numbers, as this facilitates the immediate recognition of a host as an MH. If the organization IP address space is heavily populated, it is also possible for the MHs to have their own network number. Either of these cases is an optimization, and is not *required*.

A key issue in developing protocols for mobile internetworking is which network layer(s) should provide transparent mobility. Our primary objective is to hide mobility from applications that do not wish to know about it, hence nothing above the transport layer should be changed. The choices are therefore transport, network/internetwork, datalink (or some combination). We believe that this functionality belongs to the network and internetwork (IP) layer. The reasoning is simple: the purpose of the network layer is to provide uniform access to network interfaces, and to handle routing and internetworking. It hides the different hardware addresses of the datalink layer, or the exact location of a host on an internetwork, and provides the abstraction of a network address to the layers above it. The problem with this approach is how to change the location of a host in an environment that depends on network and subnetwork addresses for routing and still maintain addressability. This paper addresses exactly this problem, and our solution is the dependence on the MSSs, and added functionality in the IP layer.

Another approach would be to dynamically change the logical (network) address of an MH to an address local to the segment of the network to which it moved. This creates a variety of problems. First, such transient IP addresses would have to be dynamically assigned and reclaimed. Next, name servers would have to be updated to reflect the change of IP addresses (assuming we want to preserve at least the host name of the machine; network users associate people with particular machines). In addition, transport-layer and higher protocols would have to be informed of this address change. While it may be sufficient for connection-oriented protocols, such as TCP [Pos81b], to inform only the transport-layer code, other protocols that do not keep the address of their peer or peers as part of their state, would fail. Every higher-level application would have to be notified every time an MH changes location. The network address of a host is used in too many places to make it easy to modify it without affecting almost every networked application.

Conversely, one could descend the network hierarchy instead of climbing it, and attempt to have the data-link layer handle moving hosts. Regardless of how this is implemented, when two MHs that are not on the same connected network, some form of encapsulation will have to be used in order to deliver packets (as the data link layer cannot interact with routing). Furthermore, the agents responsible for handling mobility would have to exchange information on the location of each MH, and that, too, cannot be done at the data-link layer if the agents are not on the same connected network. To exchange MH or other routing information, a data-link driver would have to use network-level packets to route its requests to the remote driver. In other words, some of the functionality will still be in the network layer. Finally, a new driver would have to be written for every different network interface to support mobility.

In conclusion, it is evident that functionality pertaining to mobility should be added at the network layer. As it turns out, from an implementation point of view, some modifications may be needed in the data-link layer to deal with address-resolution[1] issues. More about this in Section 2.4.1.

Lastly, 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.

2.2 Overview of the Protocols

Let us now consider the infrastructure necessary to support *intra*campus mobility. All the hosts in the campus may be on the same connected network[2] or there may be subnets

[1]mapping logical (network) addresses to physical (data-link) addresses.

[2]Connected Network: A network to which a host is interfaced is often known as the "local network" or the "subnetwork" relative to that host.

connected to each other with level-2 bridges or level-3 gateways. For each such section of the network to which MHs attach, we need to have at least one MSS.

Figure 1 shows a prototypical campus. The backbone is on subnet n0, and there are four more subnets, n1 through n4 connected with gateways gw1, gw2 and gw3. All MHs are on subnet n9. MSSs s1 through s4 define the four (wireless) cells c1 through c4. s5 is technically in c1 and links n2 to the rest of the campus network. MHs m1 through m6 are in the pictured cells, with m4 being in the wired cell supported by s2. Regular hosts and gateways have MSSs as their gateways to subnet n9. Note that MSSs have their regular IP address on the wired network, a mobile IP address for their wireless interface (if any), and a second, mobile IP address for their wired interface.

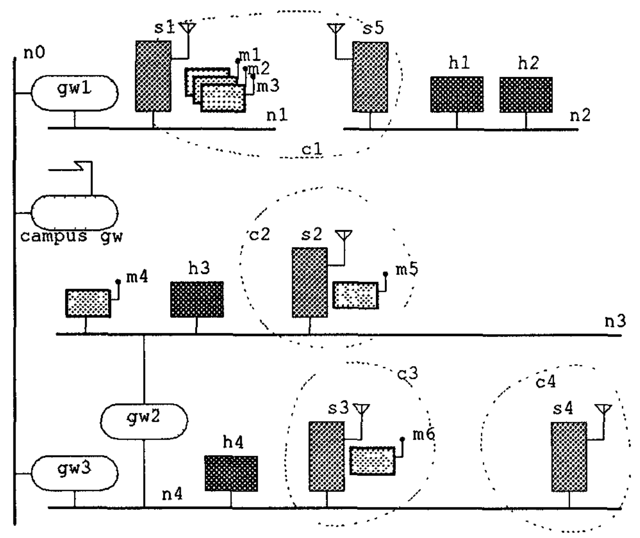

Figure 1: Prototypical campus environment

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 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.

However, these terms can cause confusion, and therefore we use the term "connected network" in this document. (Quoted from RFC1122 [Bra89])

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[3]. 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.

Another mode of operation is when an MH not only changes cell, but also switches to another interface. This

[3]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.

can happen when, e.g., an MH moves from a wireless cell to location with Ethernet, and wants to take advantage of the higher bandwidth. The MH will now receive the MSS's beacon over a different interface; it will respond to this beacon with a packet having the address of the new interface as its source IP address, but giving the MH's home address in the packet. The only difference between this set-up and the previous cases is that the MSS uses the new interface when communicating with the MH; as far as the rest of the network is concerned, the MH still has its home address.

Finally, there is an interesting special case to consider: an 'island' network, `n2`, may exist, which is an 'extension' of another subnet of that campus (in the sense that it has the same subnet number). In this particular example, it is connected to the rest of the network over a wireless link. Hosts in `n2` are considered MHs by `s5` and `s1`. Traffic to them reaches `n1` through normal routing mechanisms, is picked up by `s1`, tunneled to `s5` and finally delivered. Traffic from them will first go through the `s5`-`s1` pair, and then be routed to the rest of the network. This feature is useful when setting up temporary networks (e.g. moving a number of machines for a demonstration), or in more permanent cases when part of an organization is in a different geographical area but it is desirable that all its traffic appear to be going through the main location.

*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`.

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. The MH still has its home address; it is only using this new transient address for local delivery of packets to and from its MSS. 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

Our protocols are not affected by the nature of the physical layer; the only requirement is that link-level broadcasts are supported in order for the MSS's beacon to be received from all the MHs in a cell. This is, of course, irrelevant in the case of point-to-point links such as SLIP [Rom88] or PPP [Per90].

The cell of a *wired* interface is defined by the reach of the cable (coaxial, twisted-pair, fiber) it is connected to. The presence of an MH in a cell is strictly a function of whether its interface is physically attached to the cable. Conversely, the cell of a *wireless* interface is the geographical location around the MSS, and is defined by the propagation characteristics of the electromagnetic waves at the operating frequency. Infrared communications, for example, stop at the walls of the room, while radio-frequency communications (typically at the 1GHz band in 1991) have much more complicated propagation characteristics. In the latter case, if more than one RF cells is present, it is almost impossible not to have overlapping cells if continuous coverage is to be achieved. There, the ability of the hardware 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. In addition, 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 kinds of interactions that take place in the data-link layer are of interest to us: mapping logical addresses to physical addresses, and dynamically acquiring IP addresses when moving to a foreign campus. An example of the first is the Address Resolution Protocol, ARP [Plu82]. No extensions to the protocol are required; however, modifications should be made to the ARP code in the MSS in order to properly support mobile hosts. The second is needed when an MH migrates to a foreign campus and has to dynamically acquire a local IP address.

2.4.1 ARP Algorithm Changes

When a host needs to find the hardware address of another host in a broadcast medium such as Ethernet, it broadcasts an ARP request with the IP address of the target in the packet. All hosts on the connected network will receive this broadcast; only the one with the requested IP address will respond, unless some host has a "published" entry in its cache, in which case it too will respond.

In the context of mobile networking, we are faced with the following problem: when an MH wants to communicate with another MH, and the two MHs have IP addresses in the same subnet, the first MH will send an ARP request for the address of the second, since it thinks the latter is on the same connected network. If the second MH is indeed in the same cell, then it will respond to the ARP request. If it is not, the MSS will 'proxy-ARP' for that other MH. When that happens, the first MH will send all traffic destined for that other MH through its MSS, which will now encapsulate it in an IPIP datagram, and deliver it to the remote MSS handling the second MH's traffic. Figure 2 shows the pseudo-code for the ARP reply code in an MSS.

```
if target is not an MH then
    use the regular ARP code;
else if source is a remote or unknown MH or
        both source and target are local then
    drop the packet;
else if target is unknown then
    attempt to locate it;
else
    proxy-ARP for target;
```

Figure 2: Modified ARP algorithm

The MSS on that connected network will receive the request and respond to it. The 'attempt to locate it' step needs some explaining. As described in section 2.7.2, the MSS will contact the other MSSs in order to find which of them is handling the target MH's traffic. The host that sent the ARP request will continue ARPing until it gets a reply; if the target MH exists, a reply will arrive; on the next ARP request, the MSS will proxy-ARP. If the target MH does not exist or is unreachable, the behavior of the system is the same as when trying to locate a non-existent host on a regular network.

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.[4]

[4]If the MHs in a campus already have their own network number, the MSSs are using addresses on *that* network to transmit their beacons and handle their MHs. Thus, as far as the MHs are concerned, the address of their home network is their own.

This can be implemented in a variety of ways, all of which involve mapping an identifier uniquely associated with the mobile host (e.g., its ethernet (or other network) hardware address) to an IP address. A number of protocols, such as RARP [FMMT84] and BOOTP [CG85] provide such physical-to-logical address mappings as part of their functionality, and they can be easily extended to dynamically assign transient IP addresses.

The exact algorithms for dynamically assigning IP addresses are beyond the scope of this paper. Any number of existing and under-development protocols, such as the Dynamic Host Configuration Protocol [Dro91] may be used.

2.5 The Network Layer

In Section 2.2 we sketched how datagrams are sent from and delivered to MHs. No modifications are needed to the IP layer of the MH. As the MH switches cells, it merely changes its route table entry for the default gateway to be the MSS of that cell. A system program listening to the MSS's beacon handles that.

On the MSS, the IP layer must be modified to decide how to route a datagram. An entry is kept in the kernel for each MH the MSS knows about. The key field in each entry is the MH's home address. Additional fields include: for local MHs, the IP address to be used when communicating with the MH (if it is not the same as the home address); for remote MHs, the IP address of the MSS handling the MH's traffic; a timer field, reset every time the entry is accessed, that is used to expire entries which have not been used recently.

The output routine of the IP layer in an MSS needs to be modified; it includes the following control logic:

```
if target is not an MH then
    deliver/route using the regular IP code;
else if target is a local MH then
    deliver locally using the appropriate interface;
else if target is remote and known then
    encapsulate in an IPIP packet and
    deliver to the corresponding MSS;
else
    attempt to locate MSS and drop packet;
```

Figure 3: Modified IP output routine

IPIP, the protocol used for the encapsulation, and MICP, the protocol used to locate the other MSS, are described in the following sections. Again, note how the IP layer only needs to attempt to locate the MSS for the MH it is trying to contact; if it cannot be found, the host that originated the communication will eventually give up. This is similar to the modified ARP algorithm described above.

2.6 IPIP: 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 for virtual point-to-point connections, as already shown in Figure 3. Again, the problem we are trying to solve here is how to deliver an IP datagram to a host (the MH) whose location cannot be deduced from its IP address. In summary, here are the obvious possible solutions:

- Change the IP address of the MH as it moves. We have already shown why this is undesirable.
- Maintain per-host, rather that per-subnet, routing entries in all routers, updating them as the MHs move. This can be prohibitively expensive, both in terms of number of messages and network bandwidth needed to propagate such routes, and in terms of storage needed on the routers themselves.
- Source-route [Pos81a] all packets destined for an MH to a router on the same connected network as the MH. The problem here is how to find the location of the MH. There can be a central machine registering all moves and keeping track of each MH's location (some kind of 'network identifier' is still needed in each cell), or there can be a set of MSS-like machines keeping track of MHs.
- Explicitly define cells supported by MSSs as described in the previous section, have them keep track of MHs, and tunnel packets from MSS to MSS using either (loose) source routing or IPIP encapsulation. In a sense, IPIP encapsulation is equivalent to IP loose source routing. The main difference is that source-routing depends on an IP option being implemented on all routers along a path, whereas encapsulation only depends on a protocol module being implemented at the two endpoints of the tunnel.

Given the environment we are targetting our protocols for (both intra- and inter-campus communications, in networks using hosts and routers from a large number of vendors, and over which we have no control), we chose the last solution.

When an IP datagram reaches the output routine of the IP layer, the routing and MH tables are consulted. As described in Figure 3, if the destination MH is in another cell (with a known MSS), it is encapsulated in an IP datagram of type `IPPROTO_IPIP`, IP protocol number 94 [Rey91], and sent to the remote MSS. Thus, the IP destination and source addresses of the datagram can be considered the two endpoints of the tunnel.

Upon receipt by the remote MSS, the IP code hands the datagram to the IPIP protocol module. The latter strips the IPIP header and, after decrementing the time-to-live field feeds the packet (which now has the real destination and source IP addresses) back into the IP queue so that it can be delivered. In addition, the code checks whether the source of the IPIP packet was another MSS. These interactions are detailed in section 2.7.2.

2.7 MICP: 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 MSSs that they have changed cells. MICP datagrams are encapsulated in IP datagrams of protocol type `IPPROTO_MICP`, IP protocol number 95 [Rey91]. There are three classes of MICP traffic; MH-MSS handshakes, MSS-MSS exchanges to distribute MH location information, and MSS configuration exchanges.

2.7.1 MH-to-MSS Handshaking

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 MHs already in the cell can tell that they are still within the realm of the MSS.

It is conceivable that an MH may receive beacons from more than one MSS at a time. This is usually the case where adjacent cells of radio networks overlap. If the hardware gives an indication of the signal strength, and the MH is within the overlapping area of two cells, the MH may want to switch to with the cell that has the strongest signal. Otherwise, the MH may wait until it no longer receives the beacon from the previous MSS before it switches to the new one. The exact behavior in the presence of multiple beacons has yet to be investigated.

The beacon is an MICP packet of type `MICP_BEACON`, containing the IP address of the MSS's primary interface and the subnet mask of the current cell. It is broadcast periodically on the local broadcast address of the cell.

When an MH receives a beacon packet that is different than the previous one it received (i.e, when an MH switches cells), it responds with an MICP packet of type `MICP_GREET`. This packet contains the home address of the MH, the IP address of the MSS of the previous cell it was in (or zero if this is the first cell the MH is entering), and a timestamp.

When the MSS receives a greeting packet, it responds with an MICP packet of type MICP_GRACK (greeting acknowledge). It updates the relevant system structures to indicate that the MH is now local, and records the information supplied by the MH. It also sends a forwarding pointer to the MSS that was previously handling this MH, and whose address it got from the greeting packet. Upon receipt of the MICP_GRACK packet, the MH changes its routing tables to route all packets through the new MSS, and now considers it its new MSS.

Beacon packets should be broadcast often enough that missing some will not cause the MH to assume it is no longer in the cell. If the greeting packet is lost, the MSS will not respond to it, and the next beacon packet will cause the MH to send a new greeting. If the acknowledgement packet is lost, the MH will send another greeting until it gets an acknowledgement. Within the same cell, the effects of all three packet types are idempotent: after the first successful beacon/greeting/acknowledge handshake, duplicate beacons are ignored. Duplicate greetings cause the MSS to overwrite the MH information, and duplicate acknowledgements have no effect since the MH can tell they are coming from its current cell.

2.7.2 MH Information Dissemination

When an MSS receives the greeting, it also sends an MICP packet of type MICP_FWDPTR (forwarding pointer) to the MSS previously handling that MH. The previous MSS uses this information to properly handle any traffic for that MH originating in the segment(s) of the network it is on.

The MSS will keep retransmitting the forwarding pointer until it gets an MICP packet of type MICP_FWDACK (forwarding acknowledgement). This is needed in order to make certain that the previous MSS will not attempt to handle traffic for an MH that it no longer serves.

After an MH moves away from a cell, its MSS will continue receiving IPIP-encapsulated datagrams for it from other MSSs. It knows where the MH migrated, so it informs those MSSs of the fact by sending them an MICP packet of type MICP_REDIRECT.

When an MSS receives from another MSS 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 thinks it should have gone. If the MH had moved again, this 'courtesy' delivery will result in a new redirect being sent. Eventually, all MSSs involved in a particular MH's traffic will either converge to point to the same MSS, or their information will expire.

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 danger 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; since if another packet is received, another redirect will be sent. Eventually, one will reach the offending MSS. Hence, no explicit acknowledgement of the redirect packet is needed.

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 notifies the sender with an MICP packet of type MICP_EXPIRED. The sender will subsequently go through the peer MSS discovery process described next. The loss of this packet will result in a similar scenario to the loss of a redirect; thus, no acknowledgement of this packet is needed.

When an MSS is asked to deliver a packet to an MH it does not know about, it must discover which MSS is responsible for it. It drops the packet (again, a higher-level protocol will timeout and retransmit it) the packet and then sends an MICP packet of type MICP_WHOHAS to all the MSSs on the campus, asking them to respond if they are responsible for that MH. The MSS currently handling the MH will respond with an MICP packet of type MICP_IHAVE. This on-demand discovery of peer MSSs allows them to know only 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.

If two MSSs are in the process of a forwarding pointer exchange and they receive an MICP_WHOHAS request, they may both reply, claiming that they handle the MH. If the wrong MSS is selected, the next packet tunneled through that MSS will result in an MICP_REDIRECT being sent. To avoid this extra traffic, the MICP_IHAVE packet also includes the timestamp sent by the MH in its MICP_GREET message. This timestamp is saved, sent with MICP_IHAVE messages, and compared against the timestamp in received messages. Note that the timestamp from an MH is only compared against other timestamps from itself. At no point is the timestamp compared to a reference time base, hence the issue of keeping the clocks of MHs synchronized does not arise. All that is required of the timestamp is that it increase with time, that its maximum value be large enough that no wrap-arounds occur while the MH is in use, and that its granularity be small enough that the MH is not likely to change cells while the timestamp value remains the same. A 32-bit timer counting seconds is thus sufficient.

The MICP_WHOHAS request need not be sent individually to every MSS in the organization. If multicasting [Dee89] is supported, then the MSSs can all be in the same host group and the request can sent to the multicast address of the host group. Alternatively, each MSS may keep a list of neighboring (by some metric) MSSs, and propagate the 'who has' request to all its neighbors (except the one it received the request from), until the MSS that actually handles the MH is reached if one exists. This is essentially flooding.

If either the MICP_WHOHAS or the MICP_IHAVE packet is lost, the process will repeat when the MSS retries to send the IP packet that caused the lookup.

2.7.3 Discovery of All MSSs On-Campus

The cleanest way to send 'who has' requests to MSSs is for all MSSs on a campus to support multicast addressing and be members of the same host group, as described above. 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, as this would require that they know and support these new protocols, 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. This is accomplished with MICP messages of type MICP_NEWMSS and MICP_MSSLST.

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. The easiest such case would be to use an agreed-upon name for the central repository of MSS information, use the domain name server to acquire its IP address, and download the relevant information from the repository.

2.8 Higher-Level protocols

Layers above the transport protocol need no modification to take advantage of mobility; our protocols is to provide transparent network access to everything above the network layer. There may be some minor problems, though. TCP (and other connection-oriented protocols) has the concept of sending 'keep-alive' packets This may adversely effect TCP streams to and from an MH that stays away from an MSS for longer periods of time than the keepalive mechanism would allow. Such timeouts are usually of the order of a few minutes, thus cell switches will not affect them, but prolonged absences might. Thus, the use of keep-alives is discouraged in the presence of mobile networking.

Our protocols attempt to hide host mobility; however, it may be the case that high-level software can benefit from knowledge about relocation. For example, access to widely replicated resources, such as system programs and libraries, should not have to go through the entire IPIP process if there are copies of those resources available locally. There is on-going research at Columbia University [TD91] on file systems for mobile hosts.

3 Evaluation

As of mid-April 1991, a first implementation of the protocols exists under Mach 2.5. It consists of approximately 1200 lines of user-level code and 1000 lines of kernel code. In this section we shall give a mostly qualitative analysis of the complexity of the structures and algorithms involved based on this implementation, and the expected performance of our protocols.

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. If this is sent as an IP packet on an ethernet-like medium, the total header length is 34, for a total packet length of 46. 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. In our implementation, the observed cell switch time is under one second.

More interesting is the impact of a cell switch on the network. Two packets are exchanged between the MSS and the MH, and another two between the MSS and the previous MSS the MH was talking to. Other MSSs may have also been involved in the particular MH's traffic, either because the MH was talking to mobiles in other cells, or because it was talking to machines gatewayed to the mobile subnet through other MSSs. Each one of those MSSs will need an MICP_FWDPTR/MICP_FWDACK message pair to learn about the new location of the MH. Therefore, the total message count for a cell switch is determined by the number of connections through distinct MSSs. Also, since these remote MSSs are informed of the move only when they try to communicate with the MH, updates oocur on a demand basis, and they do not incur a burst of traffic for every cell switch.

Knowing the cell switch time is is useful because it puts an upper bound on the frequency with which a mobile may be switching cells, and hence gives us an indication of how small a cell can be as a function of the average speed of an MH.

MSS Load: An MSS should be fast enough to process all of its cell's traffic, 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 may be 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 tunneling code can find the required IPIP information. If the address space of the MHs is sparse, hash table lookups will be better suited.

Locating the information for an MH is not time-critical as far as proxy-ARPing is concerned, since this must happen only once. On the other hand, these tables constantly have to be looked up when encapsulating packets in IPIP. There is a clear memory-speed tradeoff. If we simply keep a table indexed by the MH's host part of the address, the amount of memory needed is not prohibitively large. Even if 16-bit host parts are allowed (the entire host part of a Class-B network, which translateds to 64K hosts), at 12 bytes of information per mobile host, three quarters of a megabyte are needed.

IPIP Overhead: Encapsulating IP datagrams in an IPIP packet involves prepending them with a new IP and IPIP header. Since this header changes only when the corresponding MH moves, the system can store the entire header and keep a pointer to it[5]. 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.

As discussed in the previous item, an extra pointer may be saved in the per-MH information pointing to such a prefabricated IPIP header; this header may be created the first time encapsulation through a particular MSS is used, and reused as necessary.

MICP Time and Space Complexity: There are three kinds of MICP traffic: MH-MSSMICP messages, MH location information distrubution, and MSS configuration traffic. The third 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, sends a forwarding packet to the MSS previously handling the MH, and awaits a reply. This 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. 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 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.

Audit trails of the network traffic through an MSS can serve to demonstrate this coherence. If coherence is indeed observed, it can be used as a hint to caching algorithms on which IPIP headers to keep in fast-access data structures. We expect this to be the prevailing case when most of the users in a cell are working on the same topic. An example of this is students in a classroom, or attendees of a

[5]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.

presentation, looking up the lecture transparencies as they are being shown or referenced.

In addition, it may be advantageous to cluster tunneled IP traffic, so that multiple (small) IP packets are tunelled using a single IPIP packet. This may be preferable when traffic is heavy, the destinations of most IPIP packets are the same, or the channel is being used near its maximum capacity. This clustering is used in protocols such as DEC's LAT. On the other hand, if no coherence is observed, then all packets should be treated the same way and the savings will be in the complexity of the implementation.

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, and we have already evaluated this case.

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 a network *without* these additional hosts; our approach fares well in this comparison.

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 are still routers that do not correctly process all IP options, especially source-routing.

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, and control is much more centralized.

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 the superior approach is for a site in the tree, upon next access, to update the forwarding pointers of all other sites in the tree, thus collapsing the tree to onle 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 likely to be the most common case, and because intercampus mobility 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. Our preliminary implementation and qualitative analysis shows that our protocols incur little overhead, integrate easily into a campus environment, 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.

[Bra89] R. T. Braden. Requirements for Internet hosts - Communication layers. RFC 1122, October 1989.

[CG85] W.J. Croft and J. Gilmore. Bootstrap Protocol. RFC 951, September 1985.

[Dav91] Dwight B. Davis. Wireless LANs broadcast their benefits over cable. *Electronic Business*, pages 58–62, May 1991.

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

[Dro91] R. Droms. Dynamic Host Configuration Protocol. Network Working Group, Draft, March 1991.

[FMMT84] R. Finlayson, T. Mann, J. C. Mogul, and M. Theimer. Reverse Address Resolution Protocol. RFC 903, June 1984.

[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.

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

[Per90] D. Perkins. Point-to-Point Protocol for the transmission of multi-protocol datagrams over point-to-point links. RFC 1171, July 1990.

[Plu82] D. C. Plummer. Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware. RFC 826, November 1982.

[Pos81a] Jon Postel. Internet Protocol. RFC 791, September 1981.

[Pos81b] Jon Postel. Transmission Control Protocol. RFC 793, September 1981.

[Rey91] Joyce Reynolds. Personal e-mail communication. April 19, 1991.

[Rom88] J. L. Romkey. Nonstandard for transmission of IP datagrams over serial lines: SLIP. RFC 1055, June 1988.

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

[TD91] C. D. Tait and D. Duchamp. Service interface and replica management algorithms for mobile file system clients. Technical report, Department of Computer Science, Columbia University, May 1991.

[WPD88] D. Waitzman, C. Partridge, and S. Deering. Distance-Vector Multicast Routing Protocol. RFC 1075, November 1988.