Secure Mobile Networking
Third Quarterly Report

James R. Binkley and John McHugh
Portland State University

9 July 1996

Project Status Overview

We have made a considerable amount of progress this quarter. We are in the final stages of completing a Mobile-IP implementation for FreeBSD and hope to make an alpha release to interested parties this summer. The Mobile-IP system currently contains a number of security features, including timestamp replay prevention code based on NTP and security logging, plus all of the basic security association relationships possible in Mobile-IP. We have deployed Mobile-IP in two buildings and a number of students and faculty are making use of it. Our implementation of Mobile-IP will serve as a basis for our security research.

We will begin to develop and integrate IPSEC mechanisms with Mobile-IP this summer. Our goal is a first cut secure mobile infrastructure that supports the notion of virtual secure enclaves via routing and in some sense glues Mobile-IP and IPSEC together. Our ideas are based on the analysis of Mobile-IP security that John McHugh and Jim Binkley presented in the previous report. Policy mechanisms will depend on knowledge of location courtesy of an enhanced Mobile-IP, and secure tunnel routes using the combined AH+ESP transform in the kernel. We may be able to complete this work this summer, but it may continue into next fall as there is a lot to do. We will make an informal presentation of our current design notions later on in the report.

We have gone through a round of planning for redundancy work next year and will make a short report on that subject later in this document.

As of this summer, we are bringing on board one full-time staff programmer and 3 graduate Research Assistants, two of whom are just starting the Master's program at PSU. We hope they will be with our program for a while.

We have established a Mobile-IP infrastructure in two buildings on campus. There are three agents (1 HA, 2 FA) in our PCAT engineering building and one Foreign Agent in the Mill Street CS Lab building. Three graduate students, 4 professors (3 CS, 1 EE) and two staff members have mobile laptops. Over summer, and during the next year, we hope to build up our lab with a machine or two and extend our mobile wireless infrastructure to more buildings on campus. Our goal with the latter effort is to bring wireless laptops to the numerous buildings where CS and EE professors teach. PSU is located in downtown Portland and includes a number of “white elephant” buildings that currently lack network connections in the classrooms. Bringing a wireless laptop to the classroom allows faculty to demonstrate programming concepts with real code and real running programs, which ironically up to now has been difficult if not impossible to do.

Mobile-IP Status and Planning

We have a few things left to do to Mobile-IP before we make a release. As we mentioned in the previous report, in our opinion Mobile-IP is a prime example of design by committee; i.e., there are too many features, requirements, and optional parts. Our implementation has attempted to minimize kernel involvement and maximize application daemon code. We have only implemented the use of Foreign Agents and neglected a number of other features (some required); e.g., we are not trying to implement the MN as a “do-it-yourself” FA, we have neglected to implement soft state in tunnels, or any other kind of tunnel mechanism except for the required IPIP, or the nonce replay option. We are currently NOT planning on doing much more other than cleaning up some hastily written code and working long term on improving the hand off mechanism. Even so, we have over 10000 lines of C source for the daemons. Our goal at this point is to use Mobile-IP and to begin to modify it to make all network traffic secure.

We have five executables for Mobile-IP:

1. a simple key generation program, md5gen, that spits out a 128 bit key. We hope to improve its “randomness” in the future. We understand that newer versions of the freebsd kernel (newer than the current release) will have a device in it that can be used for random number generation and intend at some point to look into that. The current application uses a number of sources of random input; e.g., “netstat -in” which provides a count of packets seen so far.

2. a mobile node daemon, mnd. It uses a configuration file /etc/mnd.conf that contains various addresses including that of the HA, and key information. Note that for purposes of our experiment we assume that laptops are single user systems, and that in fact neither our routers or laptops are ever to be multi-user. We are focused on network security, not o.s. security. Having said that, we are taking elementary security precautions; .e.g., the daemons and the utilities that query them are only runnable by root. The configuration files are only readable by root. The mnd is primarily in charge of determining the state of the mobile node, and it sets the default route to the HA or FA as appropriate. It keeps a security log file in /var/log/mnd.log. Certain events of interest security-wise are

logged there. Two kinds of security relationships may exist for the mn daemon and associated manual keys are stored in the /etc/mnd.conf configuration file. The security relationships are MN to HA (which we use by default) and FA to MN (which we have tested but are not using). (The third relationship that exists in Mobile-IP is between the FA and HA and does not apply to Mobile Nodes. We do not intend to use it either and have not as yet even tested it.)

3. a utility named mnstat that queries the mobile node daemon for information. It uses a UNIX datagram socket to communicate with the mnd so that messages from outside the machine have no way to get to the daemon. mnstat allows a user to do the following:

    (a) determine the agents available locally. Eventually this information will include wavelan signal strength.

    (b) manually command a hand off to a particular agent (and override the current choice if desired. This is extremely useful for testing).

    (c) determine the current state of the mn daemon; i.e., ATHOME (at the HA), AWAY (at a FA), and NOWHERE (no agents found).

    (d) print information of a statistical nature; e.g., how many beacons have been received.

    Basically Mobile-IP requires an authenticated and sequenced UDP request and reply. As a result, we have been able to teach mnd how to determine that connectivity with the HA has been lost. When mnstat shows the state, it can indicate that the connection to the HA seems to have vanished.

4. a Mobile-IP daemon, mipd, that runs on both FAs and HAs and is capable of being mobile out more than one interface. It uses a configuration file /etc/mipd.conf that may contain key relationships for keyed md5 as appropriate. It is assumed that the mobile router's console is secure and that this daemon and its config file are accessible by root only. The daemon maintains a visitor list and sets routes appropriately. If it supports the HA function, it sets tunnel routes via a virtual tunnel device to the MN. For a FA function, it sets a link-layer route to the MN. The latter allows direct connection between two different IP subnets; .e.g., we typically set the wavelan card on FAs to use a private net 10 IP address. Our MNs and HA are all on the subnet 204.203.71.192 (netmask 255.255.255.192). Thus the connection between the FA and MN breaks the IP subnet rule.

5. a utility called mipstat is available on agent routers to query the current status of the agent daemon (mipd).

We have attempted to minimize the kernel modifications. They consist of the following:

1. We added three new kinds of routes which primarily affected a few core routing modules and added a virtual tunnel driver. The kernel now supports two kind of "foreign" route, one for a default used by the MN, and one used by FAs to link directly to an MN. The foreign route may or may not be on the same subnet. If the subnet is not the same, information must be supplied to the kernel so that it can determine what interface to link the route to. The virtual tunnel driver (mvif0) basically puts a tunnel on an packet where the route gateway is the tunnel destination. For Mobile-IP, we run a tunnel from the HA to the FA, so the FA's Care Of Address (COA) is the tunnel endpoint. The driver basically adds an IP header to the packet with the IP src = HA, IP dest = FA, and then sends the packet to ip_output again so that it is routed normally. This driver uses code borrowed from mrouted multicast code in the kernel. A major difference with the multicast setup is that our driver ONLY handles unicast packets (multicast is left to mrouted) and tunnels are added dynamically by the mobile agent daemon (mipd). Of course, with the MBONE mrouted code, IPIP tunnels are added by hand. We believe that we have not damaged the BSD kernel's multicast functionality (in fact we fixed a bug in it).

2. UDP input is modified so that any packets coming into port 434 (the Mobile-IP port) have the interface IP address appended to them (after the UDP checksum). This is an OUTRAGEOUS hack, but it allows the mobile agent daemon to know what interface a mobile node is using and thus allows us to decouple the daemon from the kernel. It may be possible to make this a general facility say with a setsockopt so that any routing deamon could use this. This would fix a general problem with routers and the BSD stack; i.e., routers could use UDP packets (or raw packets) and thus know which interface they received a packet on.

3. There is a little bit of Mobile-IP glue code that is needed for routing and the tunnel input routine had to be carefully melded with the existing multicast input code.

4. We added a setsockopt to be used with the existing ROUTETOIF option, so that routing daemons could instruct the kernel to turn packets sent to the interface address into LIMITED BROADCAST (255.255.255.255) IP packets as required by Mobile-IP.

In terms of security features, we have implemented all 3 of the key association mechanisms as outlined above. We have also implemented timestamp (NTP) based replay attack prevention. We have tried to isolate normal error logging and security logging. State switch messages and connection messages, plus anything deemed to be a sign of peculiar behavior is logged in the security log. Fatal errors and the like go into the system log.

The most curious and hardest part of Mobile-IP has been the implementation of the timestamp replay feature. We assume that the HA has a secure source of time (e.g., authenticated NTP). In our case this is not actually true. We are running xntpd on our HA and getting the time from local routers that are hooked up to the Internet NTP time infrastructure. The MN uses what is essentially a seconds-only NTP timestamp as a security nonce (and sequence number) to timestamp authenticated packets sent to the HA. If the HA determines that the MN's time is out of whack by greater than a slop factor (currently set to 2 seconds), it NAKS the time and returns the current time (still authenticated) to the MN. The MN updates its time and tries again. The result here is that the HA is synchronizing the MN's time to seconds. If the HA receives a packet from the MN that is behind the last known time from an MN that is currently registered, a security warning is logged and the packet is ignored. It might be a time spoof, but most likely it is a delayed registration request. Note that we do NOT drop the MN from the visitor list as this would allow a denial of service attack. In general, it would seem that our IBM laptops are having their time updated only a few times a day. This mechanism could perhaps be improved by use of stable storage at the HA and by network header authentication between trusted FAs and the HA. It could also be improved by making the HA's source of time secure to the source. We don't intend to do those things. However at some point, we would like to try and use GPS to see if we can do something clever with two sources of time. (In order to do that, we will need a kernel infrastructure that can support > 1 PCMCIA card at a time).

We have tried hooking MNs up via slip connections to a remote FA connected back to our network over a modem and have noted that the time code still operates fairly well as does Mobile-IP. There are a couple of curious network topology points one can make about FAs. First of all, as a very minor security modification, we set the wavelan IP addresses for FA systems to use subnet 10, the private subnet. As a result, any host trying to access our networks through the FAs will not be successful as our ciscos refuse to route net 10 packets. Only the router itself can be attacked. Of course, the use of subnet 10 here also reinforces the notion in Mobile-IP of FAs and MNs being on different subnets. More curious is the discovery that if we take a FA home say and use it over a modem connection via SLIP or PPP, we have essentially managed to take PSU IP addresses home with us. This is either an extremely bizarre or important point (we don't know which). The upshot is that Mobile-IP could be used to allow networks at home, without requiring that internal routers actually know or support additional routes and that IP address space might be better utilized. Assume that a network provider has a class C network address and that PPP routers at home support a wavelan card with a private IP address. The network provider keeps a HA close to the terminal emulator (if not co-located). Mobile Nodes at home can be on the Internet and use the Home Address associated with the HA, and never actually be "at home" (presumably home would be on the premises of the IP provider). If the HA is close to the terminal emulator, tunnel overhead is trivial compared to circuit overhead imposed by a analog phone line or an ISDN line.

IPSEC Status

We have struggled mightily to catch up with IPSEC work and at this point have at least read the major drafts. We are sending our staff assistant to Montreal to attend the next IETF meeting. Our co-investigators are beginning to investigate how to use a formal logic system to prove (or disprove) assertions of trust about IPSEC network protocols.

We intend to use the combined AH+ESP transform (coupled with tunnels) as a major tool for Mobile-IP security. We have a concern with the combined transform and have expressed it at least to the author of the transform. We don't understand why the authentication part of his combined transform does NOT cover the IP header addresses too. His notion of supplying a nonce in the header though is interesting and we may see if we can do something with that eventually. One could point out that a notion of nonce at the IP layer requires a per key association "counter", and is hence extremely stateful. BUT if said counter is actually the time, it seems that statelessness might be possible.

For the last quarter, we have had a graduate student working with Jim Binkley, investigating the Naval Research Labs IPSEC code. Our hope was that we may be able to use it and thus save some time in the implementation of IPSEC facilities. We intend to integrate some parts of that stack into our mobile FreeBSD kernel this summer. However the parts that relate to policy-management will have to be changed as we require policy mechanism to be associated with routing, not sockets. We should point out that the NRL effort included an IPv6 (IPNG) stack, which we do not intend to use.

As a short summary, the NRL stack offers the following mechanisms:

1. a key socket which is totally analogous in nature to the current BSD routing socket (and in fact was based on it). The key socket currently allows keys to be bound to IP src and dst addresses (and spis) in a kernel key association table. Suitable keyed-md5 and des keys may be put into the kernel. We will use this mechanism.

2. AH and ESP transforms exist that basically slice a packet up just before it is shipped based on an extremely simple policy mechanism and insert AH or ESP headers before an external IP header (which may be a tunnel). These transforms are based on the original IPSEC RFCS (1826, etc.), which are now out of date. We do not intend to use ESP by itself and will change the code so that we can support basically AH, or combined AH+ESP. For a first cut, we will use the current keyed-md5 and des-cbc crypto algorithms, but we also plan to start working on a general kernel internal security driver infrastructure

that will allow both hardware (via pcmcia) and software cypto drivers to be installed and used more easily. The current des code is based on the Australian free library implementation. Over the summer, we intend to have one of our RAs, working with Sarah Mocas, implement the new HMAC-MD5 and HMAC-SHA transforms.

3. The policy mechanism in the current NRL IPSEC stack is based on two hooks. It is possible to set a global kernel base mechanism that governs all policy. This has to be compiled into the kernel at configuration time. This is an oversimplification, but for the sake of argument, you can assume that the policy is either, ALWAYS use IPSEC, or never use IPSEC. On top of this, one may also choose to bind policy to a particular socket, and in fact the socket descriptors include a simple policy description variable. So one ends up with the possibility of per socket keys and some sort of notion along the lines of "user-oriented keying". However on the downside, application code needs to be modified to take advantage of this facility and there will be no way to interoperate with existing socket/applications code that knows nothing of IPSEC. As a result of this design choice, on packet output, IP looks at the socket policy variable just before a packet is handed off to a driver. On input, the design is much more baroque. The transforms (AH or ESP) are carried out as normal layers (e.g., IP — AH — TCP — DATA and note that ESP is not intended to be nested on top of AH), and a linear search is performed to map a spi to a key table entry. A policy check must be made at the transport layer only after the TCP or UDP ports are mapped to protocol control blocks (which have socket pointers). One should point out that this is the inverse of the notion of checking policy first and then doing security transforms which was put forth in John Ioanniddis's SWIPE. We think this policy model is too simplistic, not interoperable, and more important not useful to us. We want policy to be based on routes at the network layer, which will more easily facilitate the notion of a secure virtual enclave (e.g., between a remote MN and a HA). On the other hand, there is nothing inherently wrong with binding security associations with sockets. User-oriented keying and "host-oriented" keyed are not mutually exclusive and probably both mechanisms should exist. We just want to bind policy to routes and intend to explore in that direction.

Towards A Secure Mobile-IP

The following section will outline how we intend to implement IPSEC and integrate it with our Mobile-IP implementation. We will use our Mobile-IP source base, and various IPSEC mechanisms, including the useable parts of the NRL implementation. Our theory here was presented in the previous report. The guiding principle is simply that we wish to do everything possible to "hide" (where hide means location privacy plus authentication plus encryption) data transfer between a Mobile Node and a Home Agent. As a result, for mechanism we desire to use a combined authentication and encryption transform from IPSEC (AH+ESP) and in general will nest packets in tunnels in order to hide participants and network topology from "interested" intermediate snoopers. This theory leads to some interesting routing ramifications. We do not blindly apply it to all cases. The architectural goals have three basic network topologies as a focus:

1. Network topology #1: An MN located at an untrusted FA (which may be trusted for that matter) establishes a secure network relationship that is two-way between itself and the HA. In essence, this is a virtual private network and the "circuit" between the MN and the HA is made secure.

2. Network topology #2: An MN establishes a secure link that ends at either a FA or HA, where we assume that we only want to secure the link. The network "inside" the HA or FA (seen as routers) is deemed secure.

3. An ad hoc relationship is established between two MNs that may or may not be on the same IP subnet. Eventually we may want to look at a multicast means for key distribution. The point should be made that ad hoc situations, especially when they are cut off from network infrastructure, have some interesting peculiarities that are different from generally connected networks. Key generation and distribution need not be very scalable (at least at the moment). One could manufacture a key, put it on a floppy, and pass the floppy back and forth. On the other hand, designs that assume either DNS or a general world-wise key + certificate architecture need to take into account that mobile nodes may be disconnected and disconnected for long periods of time.

Mobile Network Authentication

In order to make Mobile-IP secure, we are going to work in two areas, one of which we will call "network authentication" and the other "network security". Network authentication is going to apply to link-layer addressing and is actually a form of secure ad hoc routing. However in this first cut at ad hoc routing, we will not allow Mobile Nodes to be routers. Instead we will merely authenticate (IP, MAC) address pairs with shared secrets using keyed-md5 (or whatever the latest AH transform turns out to be). Essentially this will allow us to do two things: 1. we will not be using ARP anymore except that we will build a user interface for our systems so that if ARP is needed, it can be used, and 2. there will be no "same-subnet" else can't talk limitation.

The bottom-line here is that every host will beacon, both MNs and agents. We will augment the ICMP beacons currently used with Mobile-IP in such a way that theoretically they will still be usable with other Mobile-IP implementations, but in our case they will include extra Tag-Length-Value extensions that authenticate the information for those who have shared keys. The packets logically will contain: (the sender's IP address, mac address, a timestamp nonce, and an authentication hash). Authenticated beacons will cause link-layer routes (or ARP table entries) to be installed. Without them communication will not be possible since normal ARP will be ignored. The MN and agent daemons will time these entries out so that if beacons are not heard, connectivity will go away. One upshot of network authentication of course is that the ARP attack mentioned by Bellovin in his classic 1989 paper on TCP/IP security problems will no longer be possible. Another result is that we will be able to divide our systems from "their" systems. For example, we will know which FAs belong to us and which do not.

In order to set policy, we will create a user interface called xmnstat that will include network authentication and network security dialog boxes. The network authentication dialog will provide a set of policy choices based on knowledge of the current Mobile-node topological state; i.e., what to do when at home, when at a trusted foreign agent, when at an untrusted foreign agent (beacons have no keys or keys don't match), and what to do about link layer contact from other Mobile Nodes. Each one of these topologies will have 4 choices plus a yes/no checkbox for ARP compatibility. The choices will include:

1. never - don't install any routes period.
2. always - always install routes and ignore keys
3. key - only install routes if the keys match
4. ask - ask if ok via a popup if a mn or agent exists but doesn't have a matching key

Our default configuration will probably be "key"; i.e., we will only talk to agents and MNs that share a secret key with us. The ARP check box will basically allow or disallow ARP functionality as needed according to the topology. We will turn it off here but we might use it for reverse compatibility as needed. In terms of kernel functionality, there is currently a NOARP switch in the interface configuration flags set. We will modify the kernel so that the Mobile Node daemon can turn ARP on/off for a particular interface. On agents we will turn ARP off on mobile interfaces and of course, leave it on for true Ethernet interfaces.

Mobile Network Security

We plan on putting the NRL key socket mechanism into our kernel, and we also will implement the AH and AH+ESP transforms. However our policy work will focus on matching policy with routing.

As we mentioned above there are three topologies that we find interesting. For discussion here, we are only going to theorize about the first topology, MN via untrusted FA to the HA.

As usual, with routing there are two problems, MN to HA, and HA to MN. In the former case, we could supply in the configuration a canned list of static network or host routes. When we arrive at a untrusted FA, we would know the FA was not one of ours because of the authentication (or lack of it) in its beacons. We would install tunnel routes with an AH+ESP flag so that all packets from the MN to those sources would have the following structure:

```
| ip:  ip dst = HA, ip src = MN | AH + ESP | ip inner ...
```

As a result, we hide who the MN is sending packets to and of course authenticate and encrypt the packets. As a form of stronger policy, it would be nice if we could teach the kernel's routing mechanism how to clone packets sent to specific destinations (except for the default!, as the Mobile-IP packets sent to the FA and forwarded to the HA cannot have IPSEC applied to them). For each specific destination, we would automatically make a specific tunnel. This would represent a stronger policy choice (we might call this a "tunnel cloning" mechanism).

Sending packets back from the HA to the MN in the same form is more curious. Basically the HA already has a tunnel that almost (but not quite) reaches the MN. The HA needs to send packets to the MN in the following format:

```
ip:  ip src = HA, ip dst = FA | ip:  ip src = HA, ip dst = MN | AH + ESP |
ip inner, ip src = X, ip dst = MN | ...
```

Thus in order to hide who is sending the MN packets and thus not reveal internal topology to the world, we need to have what we will call a "double tunnel" header. The outermost IP packet would be stripped at the FA. The next header (that is still outside the crypto header), would get the packet from the FA to the MN. The MN would strip that header, deal with the crypto header, and then deliver the packet to itself. This is somewhat baroque but it meets out goal of hiding as much as possible between HA and MN.

A long-term research question here that we hope to begin to supply one answer for is: "what form should network security policy mechanisms take in a TCP/IP stack?". IPSEC and Mobile-IP only propose certain network headers, but they do not seem to take into account how those protocol requirements might be implemented (and implemented reasonably) in an operating system. The question of separation of policy and mechanism remains central. In the latest draft of Mobile-IP, there is a requirement that MNs at FAs are not supposed to use ARP, and are supposed to somehow cache MAC addresses (from the FA beacons) in order to allow the MN to talk to the FA. This is probably bad network design and it is worse kernel design. There is no attempt to deal with the central question of network authentication, which is probably crucial to Mobile-IP in the long run. With IPSEC, there has been a long term tension between "user-oriented" and "host-oriented" keying. It is not clear how policy should be architected in the kernel to supply both of those needs.

We suggest that there is nothing wrong with either of those mechanisms and that security bindings might be made to both sockets and to routes. We suspect that socket-level keys should be end to end at the application layer (or socket layer) and should not rely on network-layer IPSEC mechanisms (but they could of course use Oakley or SKIP). A real-time system might get by with one key (there might be only one protocol that accesses it). On the other hand, a multi-user system by definition needs one or more keys per user. In fact, it may be more important to worry about how to override "host" keys in some cases. For example, the current Mobile-IP manual authentication mechanism is essentially socket layer (or session layer). AH authentication between a MN and a HA, when the MN is at a untrusted FA, cannot be used as the untrusted FA will not be in the key relationship. If we blindly applied the same key (and AH) to all packets going out of a MN, we wouldn't be able to use Mobile-IP. In summary, architectural flexibility must be a long-term goal.

Theory of Mobile-IP Redundancy

In this section we are going to briefly discuss our ideas about making Mobile-IP more secure in terms of redundancy; i.e., there are three areas of work: 1. ad hoc routing; i.e., how MNs can route amongst themselves and also find paths to agents through other MNs; 2. redundant FAs., and 3. redundant HAs. Along with the Secure Mobile-IP work, this will be our focus for next year.

Ad Hoc Redundancy

Our goals here should both include both network authentication and network security. Our first cut ad hoc protocol is focused on link-layer routing and will not allow MNs to route amongst themselves. However eventually we want to develop a second-cut ad hoc routing protocol based on Dave Johnson's (CMU) ideas of source-routing; however in our case, authenticated source routing. A key idea here is that MNs should also be able to find a path to Mobile-IP agents that are further away than one hop. We also want to see if we can use signal strength as a metric and see what we can do to learn about the cons of beacons and how they might be alleviated. If we have enough time, we would also like to investigate some sort of ad hoc multicast key distribution protocol.

FA Redundancy

We find that to some extent we are already working on this as we happen to have 2 FAs within hearing of most MNs in our main PCAT building. Our current hand off algorithm is based on starvation (no beacons) and we need to teach it to be based on signal strength (if available). We can do that with wavelan and currently are working on getting the signal strength information out of the wavelan driver. We also think we can improve the hand off algorithm by imposing a test in the registration back off mechanism, so that when there is indeed greater than one agent possible, if no ACKS are heard, the system can switch and try the other agent. Since the registration is MN to HA, this means that we may be able to both deal (to some extent) with weak connectivity to an agent, and a router failure between the FA and HA.

Our long-term approach might be as follows: A mobile node would need to simultaneously register with any FAs it can hear. Mobile-IP explicitly allows that. Assume that the MNs in the presence of FAs can setup a default route to a multicast address. This means that every packet sent to default (a FA) will go to every FA without multiple resends. This is a one-way process from MN to FA. The FAs would continue to do what they do now (install a link-layer route to the MNs). Multiple registrations at the HA means that routing table entries must support a LIST of destinations. A packet sent to such a LIST route would have to be resent for every gateway in the LIST.

HA Redundancy

There are probably any number of ways one might approach the problem of removing the HA as the single point of failure in the Mobile-IP design. One HA is truly a terrible design flaw in terms of presenting an opportunity for total catastrophic system failure due to failure of one system. We suggest that for survivability reasons, it would be best to design a system where the HAs are not necessarily on the same subnet or link. They may not be far apart (although they could be), but it would be nice to have at least two HAs, where the survival of the mobile system is not dependent on one central router that leads to both HAs.

Assume that the MN has a list of HAs that it might use. It will use one HA at a time, but if it does not get ACKS from that HA, it can switch over to the other HAs in the list. Any HA will serve as "home". HAs basically have two interfaces, one wired and one "mobile". The mobile interface is a interface to a partitioned network by definition. Ironically, that means that the notion of "home" becomes blurry and the system begins to resemble FAs, and in some ways will ultimately resemble John Ioannidis's original Mobile-IP design at Columbia. HAs are paired by definition and if one of them installs a tunnel route to an MN, the other must be told about the MN and must install the same tunnel. If a link fails to a HA, normal dynamic IP routing will send packets to the other HA. This of course, may happen anyway since HA2 may be closer to a host sending packets to the MN than HA1.

Outreach

One of the PIs has met with a couple of small companies (e.g., Solution Logic) in the Portland area that are interested in network mobility and also gave a talk on the project at a recent Portland technical conference called ETACOM. Both of the PIs will be meeting with Intel security folks this summer in Hillsboro.

We have continued cooperation with OGI and one OGI Ph.D candidate on his mobility research. He is helping us this summer upgrade our kernel environment so that we can take advantage of his PCMCIA kernel infrastructure support. He is making use of basic infrastructure work on Wavelan drivers that we have done. We have submitted a joint paper with him on dynamic network interface card switching to Mobicom.

Lastly we should mention something that might be called "Inreach". The CS professors involved in this research effort are supporting an effort made by our Electrical Engineering department to start a laboratory devoted to mobile research. Of course, their research will have a focus on various hardware research topics including mobile cellular radio. However they have recognized our interest in network stacks, mobile security, and cryptography and have made the effort to include courses we teach in cryptography, security, and networking in their program. Long term, we hope that this collaboration will pay off.