

Experience with the Host Identity Protocol for Secure Host Mobility and Multihoming

Thomas R. Henderson, Jeffrey M. Ahrenholz, and Jae H. Kim
Boeing Phantom Works
P.O. Box 3707, MC 3W-51
Seattle, WA 98124-2207

Abstract—The Host Identity Protocol (HIP) is a recent protocol proposal for secure host mobility and multihoming using a cryptographic-based name space for Internet hosts. HIP aims to decouple IP addresses from transport connections in a secure manner, thereby admitting network-level mobility and multihoming solutions that do not require the use of a single IP address as a host identifier. Although HIP and related protocol proposals have been circulating for several years, there has been little reported implementation experience with the approach. This paper reports on our experience with implementing HIP and experimenting with it as a mobility management and host multihoming solution. After first introducing the HIP approach and contrasting it with other solutions, we describe our approach for implementing HIP as an extension to Linux and FreeS/WAN IPsec, including our use and extension of standard APIs. We then characterize the performance of HIP packet exchanges experimentally, and report that the computational overhead is dominated by the DSA signing of the HIP packets. Using 266-MHz Pentium II-based laptops, our HIP implementation took slightly under 1 second on average to complete connection setup, and less than 200 ms to process a mobility-initiated readdress. We also characterize the overhead due to the HIP “cookie challenge” used for stateless connection setup. We conclude by identifying areas for continued HIP development.

I. INTRODUCTION

The Internet architecture, for many years grounded in stable end-to-end connectivity, is now being forced to accommodate significant increases in wireless communication and terminal mobility. Mobility stresses the Internet in several ways, including packet routing, address management, and security. Solutions for mobility management problems have been proposed for various layers (link, network, transport, application) of the protocol stack. Of these, network-level solutions are typically the most general, because of the common use of IP.

A network-level solution to host mobility has been developed in the IETF for over six years now. Known as “Mobile IP,” it involves additional network infrastructure designed to relay packets between a “home” network and a mobile host that is identified by an invariant home address [1]. Mobile IP’s strength lies in its backward compatibility with legacy hosts, but the basic technique suffers from certain

performance, interworking, and security shortfalls that have spurred a significant body of work to extend the basic protocol approach. In parallel, the Internet landscape has changed since the original development of Mobile IP, leading other researchers to question whether additional techniques for mobility and multihoming might provide higher performance, greater security, or easier deployment. One such alternative, known as the Host Identity Protocol (HIP), was authored by Robert Moskowitz and introduced to the IETF in 2001 [2]. HIP is a new name space for Internet hosts that aims to solve some of the vexing architectural problems that the Internet continues to face, including mobility, host multihoming, site multihoming, Network Address Translation (NAT) traversal, and IPv4 to IPv6 migration.

As of this writing, HIP is considered a promising approach but there is little reported experience with an actual prototype. This paper describes our experience with implementing HIP and experimenting with it as a mobility management and host multihoming solution. After briefly summarizing the host mobility problem, we introduce HIP and describe why it is an architecturally attractive solution for secure host mobility and multihoming in the Internet. We next describe our implementation architecture and some lessons learned from implementing the protocol. Then we provide some quantitative assessment of the performance overhead incurred by using HIP. Finally, we highlight a number of areas that require further work to complete the HIP solution.

II. BACKGROUND

Host mobility causes the following four fundamental problems with the network layer in the traditional Internet architecture:

- (*Addressing*) IP routing and addressing has been hierarchically defined for scalability. As a result, mobile hosts usually find that they have a topologically incorrect interface address when they attach to a new network.
- (*Location management*) If a mobile host changes its IP address to solve the above problem, it may become unreachable to the rest of the network unless this new information is made available to other nodes.
- (*Session maintenance*) The transition to a new address can break active connections (flows) with other hosts, since transport protocols use IP addresses as part of the

connection identifier. Furthermore, extended periods of disconnection can cause higher-layer applications to abort even if underlying network handoff management operates correctly.

- (*Security*) Mobile hosts must be able to authenticate themselves to their peers upon moving, and maintain or reestablish network-level security associations.

Although dynamic address allocation problems have been largely solved by DHCP (and in the future, IPv6 address auto-configuration), location and handoff management require changes to either the network infrastructure or end hosts (or both), and solutions to security questions are still a work in progress. Various proposed solutions are centered around the natural concepts of informing correspondent hosts about mobility either directly (notification to the peer that the address has changed), indirectly (depositing location information into the network infrastructure so that it can be queried/accessed as needed), or not at all (through the use of network infrastructure such as Mobile IP home agents). One way to classify differences in proposed solutions is based on the *name space* used to provide an invariant name that uniquely identifies a host. Two candidate name spaces are the two in use in today's Internet: the IP address space and the set of fully qualified domain names. A third possibility would be to create a new name space. The IRTF Name Space Research Group is presently studying the question of whether a new name space between the network and application layers would help solve architectural strains in the Internet [3].

The proposal known as Host Identity Protocol makes the case that a new name space is needed for the Internet. By making the name space cryptographically-based (a public key), IP addresses can be decoupled from higher layer applications in a secure manner, and end hosts can be authenticated by this very name as they move. While other protocol proposals aim for similar mobility and multihoming goals (e.g., SCTP [4]), the potential benefit of HIP is that it could be applied generally to all protocols and is directly integrated with IP security protocols (IPsec) [5].

III. OVERVIEW OF HIP

As mentioned above, the HIP name space is cryptographic in nature. Specifically, the host identity is a public key, and it “signs” a particular networking stack. By making the host identity a public key, authentication of protocol transactions is automatically enabled, and the protocol is robust to man-in-the-middle attacks. Host identities can be stored in directories, such as a public-key infrastructure (PKI), or can be anonymous (in which case HIP does not authenticate the identity of the peer but can at least guarantee that an ongoing session cannot be hijacked). HIP has been designed to integrate with IPsec transport mode encryption (ESP), and after initial connection setup, incurs no per-packet overhead beyond that of ESP. The HIP handshake for key establishment has also been designed to minimize the potential for denial-of-service attacks. Figure 1 illustrates the

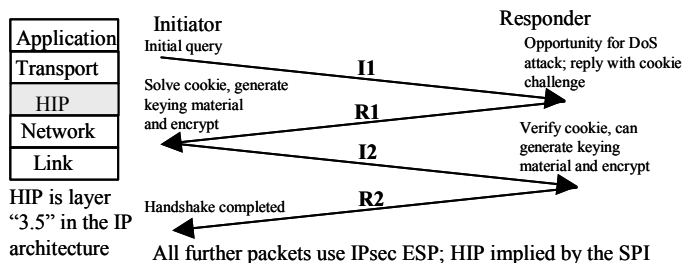


Figure 1. Brief overview of HIP architecture and four-way handshake.

HIP-enabled IP architecture, and the basic HIP handshake (which generates keying material through use of the Diffie-Hellman algorithm). Further details can be found in the Internet Drafts authored by Moscowitz [2].

As Figure 1 indicates, the key idea of HIP-enabled mobility and multihoming is to decouple the network and transport layers. By doing so, applications and transport connections can be made independent of underlying IP address changes, thereby enabling alternative solutions to host mobility, host multihoming, and network address translation. IPsec security associations (SAs) are bound to host identities, not addresses, as proposed by Bellare [6]. Therefore, the act of changing IP addresses does not break existing transport connections, nor does it trigger a reestablishment of IPsec SAs. The cryptographic host identity provides an authenticated means for correspondent hosts to notify their peers of an address change. The stateless [7] four packet handshake shown in Figure 1 is a lighter-weight version of IPsec’s Internet Key Exchange (IKE) protocol; following the establishment of keying material, no additional per-packet overhead is necessary since the host identity is implied by the IPsec Security Parameter Index (SPI), and sub-optimal routing through a home agent is avoided by mobile hosts.

Traversing multiple addressing realms is also straightforward since the NAT need only map the host identity (or SPI) to an IP address. The NAT must therefore process any HIP Readdress messages in order to update the mappings. Note that, conceptually, IPv4 to IPv6 interworking is a related problem and can be handled with a similar mechanism. At the receiving side, transport layer checksums must be recomputed (or ignored) in case readdressing occurred along the path, but this does not introduce a risk of bit corruption because the ESP decryption would fail in such a case.

IV. COMPARISON WITH OTHER APPROACHES

It is worthwhile at this point to contrast this design with other approaches. HIP-enabled mobility resembles the technique known as Mobile IPv6 with “route optimization” [8]. Mobile IPv6 with route optimization allows a correspondent host to directly route packets to the mobile host’s visited address, rather than through the home network, to improve on latency, robustness, and reduce home network congestion. It does so by maintaining a “binding cache” that matches a mobile node’s presently visited address with the

permanent home address. This mechanism is an optimization of the basic Mobile IPv6 technique of “reverse tunneling” to the home agent, which must be used whenever a binding has not been established with the correspondent host. While Mobile IPv6 with route optimization uses a Binding Update exchange to notify a peer of an address change, HIP uses a Readdress packet. Mobile IPv6 may be used with or without IPsec, while HIP is tightly integrated with IPsec, although a non-IPsec mode may be possible. Other key differences between HIP-enabled mobility and Mobile IPv6 with route optimization are the following: i) HIP does not use the concept of a home network, while Mobile IPv6 requires that initial packet exchanges between a mobile host and a correspondent host flow through the home network even if route optimization is subsequently invoked. In HIP, the location of a mobile node is obtained directly from DNS or other directory services, rather than from a home agent; ii) HIP does not incur additional per-packet overhead for carrying Mobile IPv6 Home Address or Routing Header options; iii) Mobile IPv6 can generalize to include subnet mobility (mobility of a router and its attached subnet), while HIP is purely a host-based approach; and iv) HIP inherently secures the readdressing process, while Mobile IPv6 must rely on additional mechanisms.

It is worth elaborating on the last point above. Host mobility is inexorably intertwined with security issues because inadequate solutions are prone to denial-of-service attacks. In particular, address hijacking is a key concern, since proving ownership of a particular address is difficult. Consequently, various attacks are possible on the Mobile IP Binding Update messages, used to directly notify peers that an address has changed [9]. The draft proposal for Mobile IPv6 currently centers on a mechanism to test the “return routability” of a request for Binding Update through the mobile host’s home network. While such an approach does not provide strong assurances of security, it is a solution that does not require additional infrastructure, and the door is open for stronger techniques to be added in the future. Similarly, it is difficult to securely establish in Mobile IP that a multihomed host is really the same host at a different address. Host multihoming is an important concern in future wireless tactical networks. Finally, security issues are more problematic for Mobile IPv4, where the installed base of firewalls and NATs can block operations such as reverse tunneling [10]. Researchers have proposed techniques for securing Mobile IP connections without requiring re-establishment of IPsec security associations, but this involves tunneling all data back to a security gateway in the mobile host’s home network, thereby incurring both suboptimal routing and tunneling overhead [11].

Several recent research proposals have suggested the use of a fully qualified domain name as a name space to enable mobility and/or routing across multiple addressing realms. IPNL [12] and TRIAD [13] are new NAT-based architectures that support routing by name, and resemble HIP in that they

introduce an additional protocol layer between the IP and transport layers. The architectural implications of using a domain name rather than a cryptographic host identity are discussed at length in [12], as well as the possibility of combining HIP with IPNL to provide stronger security for that approach. Transport-layer approaches to mobility (TCP-Migrate [14] and SCTP [4]) propose mechanisms, even integrated with security techniques, to allow for host multihoming or dynamic readdressing of a connection; however, HIP applies generally to all protocols and is integrated with IPsec by design. Finally, a session-layer approach to name and address decoupling (using session identities rather than host identities) is proposed in [15].

Even if architecturally attractive and deployable,¹ the question still remains whether HIP incurs too much implementation complexity and performance overhead to be useful. The remainder of this paper describes our experience with a prototype implementation of HIP, including the results of some performance benchmarks, and our view of directions for future work.

V. HIP IMPLEMENTATION DETAILS

While HIP was originally proposed with IPv6 in mind, it also can work for IPv4, as our implementation experience illustrates below. Our IPv4 HIP implementation builds on the FreeS/WAN IPsec extensions for the Linux 2.4 kernel. FreeS/WAN² provides an implementation of the Internet Key Exchange (IKE) key management protocol (which runs in user space), and implements security association and security policy databases in the kernel for insertion and verification of IPsec Authentication Headers (AH) and Encapsulated Security Payloads (ESP). FreeS/WAN has been most frequently used to support IPsec “tunnel” mode (for use in building VPNs), but recently, basic support for “transport” mode IPsec connections have been added. The latter mode is most relevant to HIP, because it is implemented in the end hosts themselves.

Figure 2 provides a basic overview of our implementation. Using FreeS/WAN as a base, we have replaced the “*pluto*” daemon (FreeS/WAN’s implementation of IKE) with a HIP daemon (named *hipd*), running in user space. Since the HIP handshake is used to establish keying material for an IPsec security association (SA), HIP can be thought of as a lighter-weight replacement for IKE, with less policy granularity [2]. “*KLIPS*” is the kernel portion (module) of the FreeS/WAN implementation. We made small modifications to *KLIPS* to allow, during incoming packet processing, a SA to be selected on the basis of SPI and host identity, rather than SPI and static destination address (thereby allowing addresses to change without reestablishing the SA). Additionally, we used the

¹ Deployment is a legitimate open question since almost all infrastructure-based services, and even many end-to-end protocols, are challenging to deploy in today’s heterogeneous, firewalled, and NATted Internet.

² <http://www.freeswan.org/>

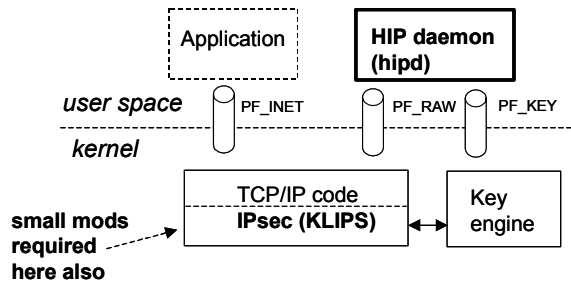


Figure 2. HIP implementation structure.

cryptographic libraries available in OpenSSL.³

As shown in Figure 2, *hipd* relies on two standard APIs to communicate. HIP is able to send packets to other hosts using the PF_RAW socket interface, thereby bypassing transport layer connections. Key management interactions with the kernel are performed using the PF_KEY API [16]. Using the PF_KEY interface, *hipd* registers with *KLIPS* to handle connection negotiations. When *hipd* receives a PF_KEY connection request from *KLIPS*, it carries out the four-packet HIP exchange and uses further PF_KEY messages to provide *KLIPS* with SA parameters and the required keying material, and to set up the extended routes. We have extended the PF_KEY API to support three other messages used for HIP: a “get sequence number” query to retrieve the current sequence number of the security association, a “get LSI” query to fetch the LSI assigned for a given IP address, and a “readdress” message to tell the kernel that the peer has readdressed its security association.

HIP exchanges produce a 32-bit Localized Scope Identity (LSI), a representation of a Host Identity intended to replace IP addresses within local system calls. To bind IPsec SAs to the LSI instead of the IP address, we added a new kernel table for the storage and retrieval of the LSI, Host Identity Tag (a compressed version of the Host Identity), and Host Identity. First, *hipd* negotiates SAs between two IPsec-enabled hosts, with the setup invoked either manually or automatically (opportunistically). Negotiations involve deciding upon which algorithms will be used, and deriving the cryptographic keys that will be used for authentication and encryption.

A. Path of a packet

Hosts begin using HIP via a system-set trap policy or with an indication from the user application (discussed further below). FreeS/WAN’s IPsec has a packet-trapping *eroute* (extended route) interface for indicating that encryption is desired for a particular destination. When a packet matches a trap, *KLIPS* will buffer the packet and issue a PF_KEY message to the HIP daemon (*hipd*). *hipd* then performs the four-packet HIP exchange with the peer machine’s *hipd*. For both ends of the connection, this handshake authenticates the host’s identity, generates the required keying material, and sets up the SA. The HIP proposal allows for data to be sent with the third and fourth HIP handshake packets (e.g., a TCP SYN

can be piggybacked on the I2 packet shown in Figure 1), but we have not yet implemented this option. Once the SA is established, the outbound packet (and any subsequent packets) are sent using ESP encryption, and the HIP association is implied by the SPI. On the receive side, the peer’s *KLIPS* module will receive these ESP packets and locate the SA using the SPI (contained in each packet’s header) and its internal HIP/LSI table. When the incoming SA is found, the packet is then decrypted and passed to the correct socket.

B. Readdress

Our HIP daemon monitors a host’s interfaces looking for changes to its IP address. When a host changes IP addresses, *hipd* detects the change, sends a HIP Readdress packet to all active, HIP-enabled sockets on that interface, and issues a special PF_KEY readdress message to *KLIPS*. *KLIPS* walks through a list of the established TCP, UDP, and RAW sockets, searching for any occurrence of the old IP address and replacing it with the new, while also updating the SA and LSI databases. The same actions are performed when a host receives a HIP Readdress packet, so that both ends of a secure connection maintain communications using the new address. As described in the draft specification, the synchronization of the readdress is coordinated through use of the ESP sequence number; the HIP Readdress message tells the peer after which sequence number the address change should go into effect.

Our implementation successfully handles a host readdress with active connections, experiencing little delay aside from the inherent delays due to ARP (and possibly DHCP) on the newly-addressed interface.

C. HIP Application Programming Interface (API)

One aspect of HIP not covered in the protocol proposals is what changes are needed to the sockets API. We envision two methods whereby the kernel can be configured or instructed to use HIP on a connection. Ideally, HIP-aware applications should be able to explicitly ask for a HIP-enabled connection with only slight modifications to their code. In our implementation, we added a socket option (*setsockopt()* system call) to flag a request to use HIP. This system call precedes the *connect()* call, and alerts the kernel that the value being passed in the IP address structure is an LSI, not an IP address. To obtain this LSI, an application can call a new resolver routine, *getlsibynname()*, in place of *gethostbyname()*, or alternatively, use the more advanced call *getaddrinfo()*. An interesting policy question is what to do with a connection that has requested the use of HIP, but for which the HIP handshake fails for some reason (e.g., the peer does not implement HIP). Additional socket option values (“must use HIP,” “should use HIP”) are possible to explicitly expose this policy to applications.

Otherwise, HIP support could be added transparently to the sockets implementation, such that legacy applications could be used, and such that decisions to use HIP could be based on policy (e.g., all non-local IP addresses) rather than explicit application requests. In this case, the kernel could do a

³ <http://www.openssl.org/>

reverse DNS lookup for the Host Identity based on the provided IP address, and determine a suitable LSI for internal use. Certain applications that use IP addresses in the application data stream (e.g., FTP) would need to be remapped just as they are when traversing NATs, if they were to be used in transparent mode above a HIP-aware kernel; hopefully the use of IP addresses in this manner can continue to be deprecated. Presently, we use unmodified IPv4 applications and impose HIP via policy decisions by setting up IPsec extended routes for a given destination.

In our implementation, we added a PF_KEY socket option to designate a socket as HIP-enabled. When the PF_KEY socket has the HIP option set and is used for adding an SA, *KLIPS* will store and retrieve the new SA using the LSI instead of destination IP address. When a readdress occurs, this table will be updated to reflect the new IP address. User applications can utilize the same socket option (for their TCP or UDP sockets) to indicate their desire to use HIP with a 32-bit LSI instead of with an IP address for establishing connections. [2]

VI. PERFORMANCE

HIP has a built-in mechanism based on the concept of a “cookie challenge” that is designed to thwart denial-of-service attacks on a server [7]. Specifically, the responder (server) issues a cookie challenge to the connection initiator (client), and is able to set the level of difficulty of this challenge based on the level of trust it has with the client [2]. The difficulty of the cookie challenge is a parameter that affects the total time required to perform a HIP handshake. During an exchange, the responder chooses the difficulty for the initiator when sending the R1 packet in response to the first I1 initiator packet (Figure 1). This difficulty is the number of bits of a hash target required to match the hash supplied by the responder, of an integer concatenated with another random integer, and must be solved via trial-and-error on the solution space. To get a sense of the impact of cookie difficulty on typical portable computer hardware, we profiled the time required to solve the cookie challenge for various levels of difficulty. We performed our measurements on two Dell Latitude Pentium II 266MHz machines, directly connected via 10Base-T Ethernet, running our HIP extensions on Linux kernel 2.4.10. We used an average of ten trials to account for the possibility that the task might occasionally be preempted by the non-real-time operating system. Figure 3 plots the results of this experiment, indicating, as expected, the degree of difficulty increases exponentially (linear in our log-scale plot) with the number of bits in the cookie puzzle. Requiring a difficulty of over 16 bits (worst-case run time of 2^{16} tries), for example, may produce an unacceptable wait time of over 1 second.

Another measure of performance is how much time is spent in the cryptography library during a HIP handshake. The HIP

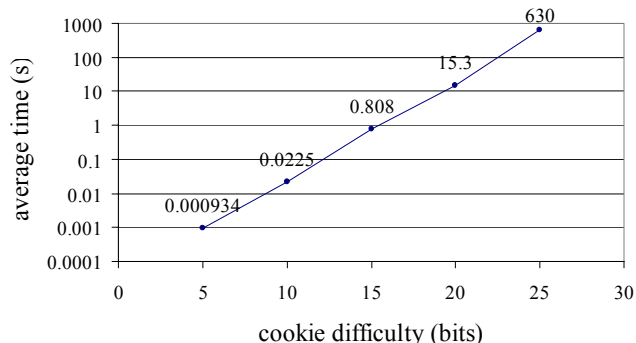


Figure 3. Average time spent solving cookie challenge.

exchange uses the cryptography library to perform Secure Hash Algorithm (SHA-1) hashing, Triple-Data Encryption Standard (3DES) encryption and decryption, and Digital Signature Algorithm (DSA) signing and verification. Table 1 presents the results of averaging the time spent executing each of these tasks, for five different HIP exchanges on the same hardware described above. Each HIP exchange takes under one second to complete with a cookie difficulty of 10. It is worth noting that the DSA signing and verification procedures for the three HIP packets occupy much of this time, typically over 40% of the total, but the relatively large overhead of software-based public key operations is well-known.

TABLE I
TIME SPENT DURING TYPICAL HIP EXCHANGE

	SHA Keymat Hashing	DSA	Cookie Challenge	3DES in I2	Total Exchange Time
Initiator	80 <i>us</i>	450 <i>ms</i>	30 <i>ms</i>	530 <i>us</i>	0.95 <i>sec</i>
Responder	110 <i>us</i>	410 <i>ms</i>	N/A	530 <i>us</i>	0.79 <i>sec</i>

Performing a readdress, like the initial HIP exchange, requires less than one second to complete. Using the same Pentium II 266MHz test bed, readdressing the responder takes roughly 130 ms on average for the responder and 180 ms for the initiator. About 94 percent of this time is spent performing the DSA signing and verification algorithms.

VII. DIRECTIONS FOR FURTHER WORK

HIP is a promising protocol architecture for more cleanly integrating host mobility and multihoming with IP security protocols, but a significant amount of experimental experience is needed to validate the approach. In this paper, we have described our implementation approach for a basic HIP prototype for IPv4 using the Linux 2.4 kernel and leveraging the FreeS/WAN IPsec and OpenSSL implementations, and have reported on initial performance results. We have identified above some minor PF_KEY and PF_INET API and resolver library extensions that would be useful for HIP. Below, we summarize directions for further work in this area.

- A longstanding problem in public-key-based solutions is how to design/deploy the key infrastructure. HIP in principle does not require a key infrastructure if everyone were to remember the keys, but this is unrealistic—a resolution from (human readable) name to host identity is needed. The HIP proposal suggests that DNS may be used to store host identities and provide a mapping to the current IP interface address(es), provided that a DNS name is available to index the host identity (this remains problematic if a host only has a host identity of the peer, but neither the IP address nor the domain name). However, there may be performance drawbacks of using DNS in this manner, particularly relating to the overhead required of DNS servers to sign and distribute their zones when they change, as well as to the heavy reliance on caching for scaling. With this in mind, Moscovitz has proposed a special “rendezvous” server that has a non-changing DNS record but which is used to relay the first HIP packet to the current address; conceptually, this resembles a Mobile IP home agent. Other solutions for mobile distributed databases for the tactical environment (e.g., [17]) may be attractive as well.
- We found that the particular structure of FreeS/WAN for IPv4 (implemented like a tunneling driver) was not the most advantageous, because it requires *hipd* to manipulate the IPsec *eroute* routing tables whenever readdressing takes place. We believe that an IPsec implementation more naturally integrated with the normal networking stack would significantly lessen the kernel modifications required for HIP, and we envision that the proposed rearchitecture of FreeS/WAN for IPv6⁴, as well as native IPsec support proposed for the Linux 2.6 kernel series, may be beneficial in this regard.
- HIP-enabled mobility and Mobile IP are both classified as “macro-mobility” solutions that suffer from undesired end-to-end latency when readdressing is rapid, such as in a cellular environment. A number of “micro-mobility” protocols, designed for integration with Mobile IP, are being designed in the IETF. We envision that such protocols could be generalized also to support HIP. Conceptually, the micro-mobility problem is not much different from traversing a NAT (where the NAT can mask the address change from the correspondent host) and therefore may be relatively straightforward to integrate with HIP. Work on validating HIP’s proposed ease of use in a NAT environment would be particularly valuable if it also considered the case of micro-mobility.
- Should HIP replace Mobile IP, or is it a complementary approach? Our view is that HIP provides potential performance, security, and addressing realm interworking benefits over Mobile IP in some scenarios, particularly heterogeneous mobile networks with multihomed hosts (such as in a tactical network). However, Mobile IP offers potential advantages when entire subnets (routers) are mobile, where HIP is not deployed, or where

infrastructure to support pervasive IPsec has not been deployed. There is also considerably more experience with Mobile IP than with HIP at this writing, and HIP would need to be fully specified to draw complete comparative conclusions. One possible evolutionary path is that HIP or Mobile IP could be selected on a per-network or per-application basis, depending on the mobility mode desired. Another path would be to use HIP to augment Mobile IP, such as using HIP to secure the Mobile IP Binding Update. We therefore believe that further work on how HIP and Mobile IP complement each other is warranted.

REFERENCES

- [1] C. Perkins, *Mobile IP Design Principals and Practices*, Addison Wesley, 1997.
- [2] R. Moscovitz, “Host Identity Payload and Protocol,” Internet Draft: draft-moscovitz-hip-05.txt (work in progress), November 2001. Available online at <http://homebase.htt-consult.com/HIP.html>.
- [3] E. Lear and R. Droms, “What’s In a Name: Thoughts from the NSRG,” Internet Draft: draft-irtf-nsrg-report-08.txt (work in progress), December 2002.
- [4] R. Stewart et al., “Stream Control Transmission Protocol Dynamic Address Reconfiguration,” Internet Draft: draft-ietf-tsvwg-addip-sctp-05.txt (work in progress), May 2002.
- [5] S. Kent and R. Atkinson, “Security Architecture for the Internet Protocol,” Internet Request for Comments (RFC) 2401, November 1998.
- [6] S. Bellovin, “EIDs, IPsec, and HostNAT,” Presentation at 41st IETF meeting, Los Angeles, CA, March 1998.
- [7] T. Aura, P. Nikander, and J. Leiwo, “DOS-resistant Authentication with Client Puzzles,” in Christianson, Malcolm, Crispo, and Roe (Eds.) *Security Protocols*, 8th International Workshop, Cambridge, UK, April 3-5, 2000; revised papers, LNCS 2133, pp. 170-177, Springer 2001
- [8] D. Johnson, C. Perkins, and J. Arrko, “Mobility Support in IPv6,” Internet-Draft: draft-ietf-mobileip-ipv6-18.txt (work in progress), June 1, 2002.
- [9] P. Nikander, “HIP, IPv6, and Mobility,” Presentation to 51st IETF meeting, August 2001. Available on-line at <http://http://www.ietf.org/proceedings/01aug/slides/hip-2.pdf>.
- [10] S. Mink et al., “Towards Secure Mobility Support for IP Networks,” *Proceedings of the IFIP International Conference on Communication Technologies (ICCT 2000)*, pp. 555-62, Aug. 2000.
- [11] M. Burton et al., “Integration of IP Mobility and Security for Secure Wireless Communications,” *Proceedings of IEEE International Conference on Communications (ICC)*, pp. 1045-1049, April 2002.
- [12] P. Francis, “IPNL: A NAT-Extended Internet Architecture,” *Proceedings of ACM Sigcomm Conference*, 2001.
- [13] M. Gritter and D. Cheriton, “An Architecture for Content Routing Support in the Internet,” *Usenix Symposium on Internet Technologies and Systems*, March 2001.
- [14] A. Snoeren and H. Balakrishnan, “An End-to-End Approach to Host Mobility,” *Proceedings of ACM Mobicom Conference*, 2000.
- [15] McAuley and R. Morera, “Name and Address Decoupling in Support of Dynamic Networks,” *Proceedings of IEEE MILCOM Conference*, 2002.
- [16] D. McDonald et al., “PF_KEY Key Management API, Version 2,” Internet Request For Comments (RFC) 2367, July 1998.
- [17] R. Jain et al., “Enhancing Survivability of Mobile Internet Access using Mobile IP with Location Registers,” *Proceedings of 1999 IEEE Infocom Conference*, vol. 1, pp. 3-11, 1999.

⁴ <http://www.ipv6.iabg.de/projects/freeswan/index.shtml>