P3

Publication P3

Pekka Nikander, Jukka Ylitalo, and Jorma Wall, "Integrating Security, Mobility, and Multi-Homing in a HIP Way", in Proc. of Network and Distributed Systems Security Symposium (NDSS'03), pp. 87-99, San Diego, CA, US, February 6-7, 2003, ISBN 1-891562-16-9, Publisher: Internet Society

© 2003 The Internet Society.

Reprinted with permission.

Integrating Security, Mobility, and Multi-homing in a HIP Way

Pekka Nikander, Jukka Ylitalo, and Jorma Wall

Ericsson Research NomadicLab

Abstract

The current trend in mobile networking is towards mobile hosts that have multiple network interfaces, e.g., WLAN and GPRS. However, when the current Internet architecture was originally designed, neither mobility nor multi-homing were considered. In the current architecture an IP address represents both a host's identity and the host's topological location. This overloading has led to several security problems, including the so called address ownership problem, making IP mobility and multi-homing unnecessarily hard from the security point of view.

In this paper we show how the Host Identity Payload (HIP), being discussed at the IETF, can be used to simultaneously solve the security problems, and many of the practical problems, related to end-host multi-homing and end-host mobility. Basically, HIP introduces a new cryptographic name space and protocol layer between network and transport layers, breaking the fixed binding between identities and locations. The approach is especially suitable for large open networks, where no pre-existing trust relationships can be assumed. We also report our early implementation experiences.

1. Introduction

When the TCP/IP protocol suite was originally designed in the late 1970's and early 1980's, it was hardly imaginable that most of the world's computers would eventually be mobile and have several distinct network connections at the same time. Thus, the protocol suite was designed with singly-homed statically located hosts in mind. In that world, the location bound IP addresses served beautifully as identifiers for the hosts, since hosts rarely if ever moved between locations.

Years ago, with the introduction of dynamic address assignment in PPP and DHCP, the assumption that an IP address would uniquely identify a host was broken, and the situation was further worsened by the introduction of private IP address spaces and NAT [1][2]. Currently it looks like that the emergence of ubiquitous computing and ad hoc networks will soon lead to a situation where the *majority* of computing hosts are multi-homed and mobile, and have no static addresses.

In addition to the nature of hosts, also the nature of users have changed during the years. For many years, the Internet was basically used by a fairly homogenous user community where everybody more or less trusted everyone else. Not so any more. Trustworthiness must now be proved through explicit cryptographic mechanisms.

In a word, the environment has changed. Looking from the 1980's point of view, the requirements for mobility and multi-homing, together with the host-to-host signalling security, are new. Addressing these within the limitations of the current architecture has turned out to be hard; therefore, it may be necessary to do some radical re-engineering for the architecture to bring the TCP/IP protocol suite in par with the new requirements. The intention of this paper is to work as a vehicle in that re-design discussion.

Many of the issues discussed in this paper are in no way new, but have been floating around for a number of years. Our main contributions stem from addressing mobility, multihoming and related security *at the same time*, and arguing how they can be handled in a fairly orthogonal way. In particular, we define an orthogonal end-host mobility and multihoming architecture, where the properties for *end-points*, parallel communication *paths* (i.e. multi-homing), *mobility*, and *related security*[1] are neatly separated into different dimensions.

The rest of this paper is organized as follows. In Section 2 we discuss the nature of mobility and multi-homing, thereby paving the way for the forthcoming discussion. Section 3 includes brief summary of the most important related work. Section 4 defines the proposed new architecture in detail, and Section 5 discusses it from the security point of view. Section 6 reports our current implementation status. Finally, Section 7 concludes this paper.

[1] To be more precise, in this context "security" means mobility and multi-homing related *signalling authorization*, i.e. access control on believing in the contents of received mobility or multi-homing signalling messages. Especially, it does *not* denote generic application-level end-to-end security.

2. Background

In this section we summarize the necessary background for the following discussion. We limit this background discussion to analysing mobility and multihoming from an end-host point of view, illustrating the resulting security problems. Later, in Section 4, we return to these concepts; there we show that they are actually quite similar and, under certain circumstances, may be taken as duals of each other.

2.1. Mobility

For the purposes of this paper, we define *mobility* to denote the phenomenon where an entity moves while keeping its communication context active (see e.g. [2]). With that we mean that an end-host, i.e. a computational unit hosting a number of communicating processes, changes its topological point of attachment. At the same time, however, we want to make sure that all active communication contexts remain active, and the processes do not see mobility other than, possibly, in changes to the actually experienced quality of service.

To reflect reality, we assume that there are a number of mobile nodes that attach to a relatively fixed network (see Figure 1). Furthermore, we assume that the network layer address prefixes are structurally determined by the network. That is, we assume that the network topology determines the routing related portion of the IP layer addresses. This assumption reflects the fact that in a large network it is important, in order to keep the routing table sizes manageable, to keep routing prefixes consistent with the network topology. Furthermore, for the sake of simplicity, the considerations for link local, site local, anycast, and multicast addresses are beyond the scope of this paper. That is, we assume that all addresses are globally routable, unless explicitly stated otherwise.

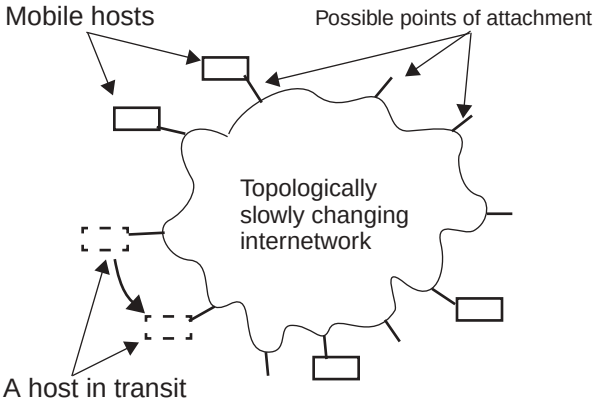

Figure 1: The mobility model

As a consequence of these assumptions, whenever a node moves, its network layer address *necessarily* changes. Thus, in order to continue to communicate, the host must be able to signal the changes in its addresses to its active peers. Furthermore, this signalling must be secure since unsecured signalling can lead to a unauthorized traffic diversion and denial-of-service attacks.

2.2. Multihoming

Multi-homing refers to a situation where an end-point has several parallel communication paths that it can use. Usually multi-homing is a result of either the host having several network interfaces (end-host multi-homing) or due to a network between the host and the rest of the network having redundant paths (site multi-homing). In this paper we concentrate on end-host multihoming.

From our theoretical point of view, a multihomed end-host is a node that has two or more points-of-attachment with the rest of the network. This is illustrated in Figure 2. This situation can be characterized as the node being reachable through several topological paths; the node is simultaneously present at several topological locations. As a consequence, it also has several network layer addresses, each of which reflects one of the topological locations. In the general case, the addresses are completely independent of each other.

2.3. The security problems

There are a number of security problems associated with mobility and multi-homing. These problems stem from the need of assigning several IP addresses to a single host, and from the desire of using these addresses in an interchangeable way. That is, in an ideal mobility and multi-homing solution the hosts can use any of their peer's addresses without having to worry about the validity of the address.

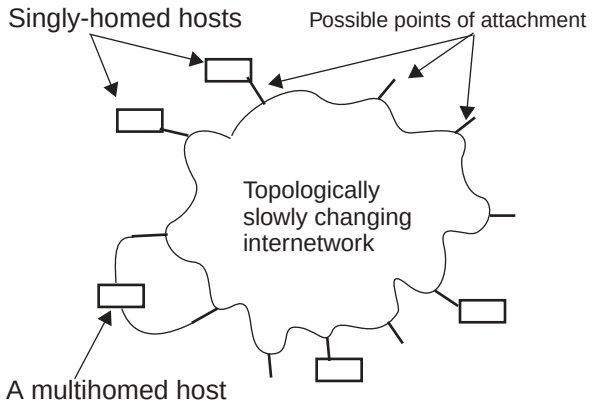

Figure 2: The multi-homing model

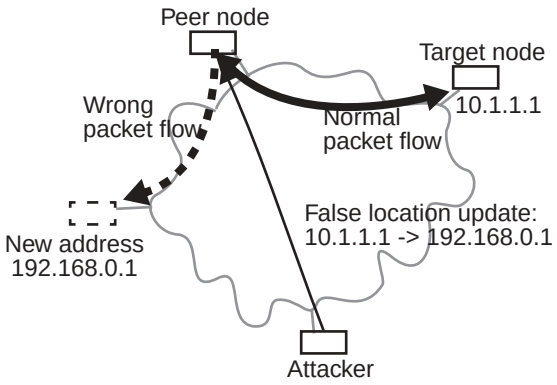

Figure 3: Address stealing attack. The peer believes that 10.1.1.1 is now at 192.168.0.1

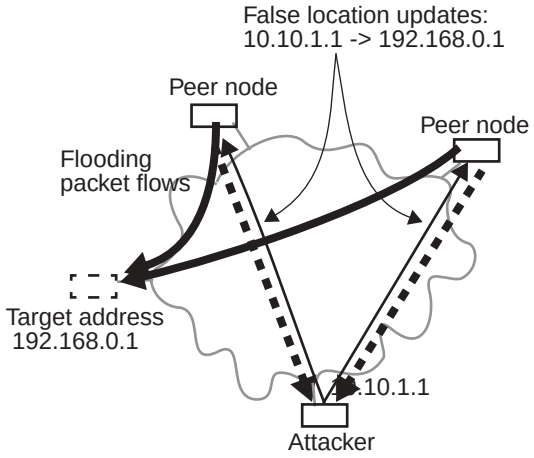

Figure 4: Flooding attack

Given this, there are two basic security problems: *address stealing* and *address flooding.* In an address stealing attack a malicious node claims to "own" an address that some other node is currently using, with the intention of launching a masquerade, man-in-the-middle or denial-of-service attack against the owner of the given address. In an address flooding attack a malicious node (or group of nodes) makes a large number of innocent peer nodes to believe that the attacker has become (better) available at a target address, causing the peer nodes to flood the target address with unwanted traffic.

Based on the mobility and multi-homing models, it is easy to see how an attacker can launch the first attack. It simply informs the peer node(s) of the target node that the target node has moved into a new address. Unless the recipient of this location update is able to securely verify that the sender of the update indeed was earlier at the target address, all the future traffic destined to the target address is directed elsewhere, resulting in masquerade, denial-of-service, or man-in-the-middle situation, depending on the other actions and the actual location of the attacker. This attack is illustrated in Figure 3.

The second attack results from failing to check that the sender of an address update is indeed reachable at the claimed new address. If a recipient of an address update blindly starts to send messages to the new address, the messages may be delivered to an innocent third party that now receives excess traffic. While one node sending bogus packets may not be that bad, hundreds or thousands of nodes sending extra traffic at the same time are likely to fill the communication link causing denial-of-service. See Figure 4.

While these security problems can be solved to an extend with reachability checking [27], it is hard to completely solve the problems within the current architecture. There is no way of checking that a node claiming to be a given address is actually the node that is indeed located at the address, topology wise. That is, when Bob is communicating with someone that Bob thinks is at an address A, Bob cannot be sure that someone is actually at A and not at some topologically intermediate point between A and Bob, or at a point close to A or Bob, thereby being able hear all the messages that Bob sends to A.

Since the IP addresses are used as the primary identifiers today, Bob cannot explicitly say that he wants to talk to Alice, but he must simply send packets to the address that he believes that Alice has. Thus, if Bob wants to talk to the Alice, supposedly at A, and ends up talking to Carol at some intermediate point, Carol can easily tell Bob that she has moved to a new address D even if reachability checks are used.

Within the scope of the current architecture, the only really secure solution would be to provide a credential infrastructure binding addresses to public keys, thereby creating the possibility of binding nodes and addresses in a stronger sense. However, providing such an infrastructure would be extremely hard due to practical reasons.

3. Related work

As we already mentioned, our work is mostly based on the HIP drafts [1][7][8], with quite a lot of influence from Noel Chiappa's and Steven Bellovin's writings [2][3], the IRTF NSRG report [4], and our work in securing Mobile IPv6 [5][11][12][13]. The real background was laid out by Jerome Saltzer in his seminal works [14][15], but he was clearly ahead of his time.

In addition, there are a number of less related existing and proposed approaches to address end-host mobility and multi-homing. In this section we give a brief overview of the other works that we know of, starting from well established approaches, and proceeding to more adventurous proposals.

SCTP. Stream Control Transport Protocol (SCTP) is an IETF proposed standard transport protocol, which may eventually replace TCP and perhaps also UDP. In it, each communication process is associated with several IP addresses. While the SCTP approach is sound as such, the proposed mobility extensions [16] are bound to be plagued with the same security problems that Mobile IPv6 was recently hit (see below). Since SCTP does not include explicit end-point identifiers, solving the security issues in a scalable way may be even harder than with Mobile IPv6.

Mobile IPv6. In Mobile IP, a static address is assigned to each node. Mobile IP does not currently address end-host multi-homing, but there are informal proposals floating around how a single mobile node could use multiple home addresses and multiple care-of-addresses at the same time [17]. Until recently, the largest unsolved problem in Mobile IPv6 was achieving a scalable security solution. The currently proposed solution is based on the ideas of relying on the routing infrastructure to check that a mobile node is reachable both at its claimed home address and its claimed current address (care-of-address) (Return Routability, RR). This approach is not very secure, even though it is claimed to be (almost) as secure as the current IPv4 internet. Thus, there are discussions going on about better proposals, e.g. hashing a public key and other information to the low order bits of an IPv6 address (Cryptographically Generated Addresses, CGA) [9][13].

Multi-homed TCP. Multi-Homed TCP [18] was a proposal by Christian Huitema to extend the TCP protocol to handle end-host multi-homing and mobility. The basic idea was to replace the port numbers (TLIs) with a single 32 bit Protocol Control Block Identifier, which becomes independent of the IP addresses used. On the security side, the work did not really address the authentication or denial-of-service problems; we believe that addressing them would have led to an architecture more or less similar to the presented one.

LIN6. LIN6 [19] is an approach somewhat similar to the 8+8 or GSE [20]. The basic idea is that each host has a 64 bit globally unique identifier, called LIN6 ID, which is present in the IPv6 interface identifier portion of all IP addresses used by the node. At this writing, the largest unsolved problems in LIN6 are related to the scalability aspects in the security side. The address update messages are protected with IPsec AH, thereby requiring some kind of global infrastructure in order to establish the required security associations.

Homeless Mobile IPv6. Homeless Mobile IPv6 [21] was an idea by Nikander et. al. of adding end-host multi-homing to Mobile IPv6, and at the same time getting rid of home addresses and much of the extension header overhead. The basic idea was fairly similar to SCTP, but the implementation was placed on the network layer instead of the transport layer. The project did not properly address the involved security problems; instead, the security considerations lead to the definition of the address ownership problem [12][22].

TCP Migrate. Snoeren and Balakrishnan [23] propose an extension to the TCP protocol that allows the TCP endpoints to migrate from an address to another. Being structurally fairly similar to Huitema's Multi-Homed TCP, the approach solves the security issue through using unauthenticated Elliptic Curve (EC) Diffie-Hellman to generate a session key, separately for each TCP session. However, they do not solve the double jump problem, and rely on Dynamic DNS for initial contact.

4. Architecture

This section defines our Host Identifier based multi-homing, mobility, and security architecture. In Section 4.1, we outline the overall architecture as a layered structure. In Section 4.2 we give the exact definitions for the required terminology and components. As it turns out, just defining the terminology in a new way naturally leads to looking at the mobility and multi-homing situation from a new point of view. That, in turn, leads to the new architecture, and provides the basic facilities for multi-homing and mobility trivially, as discussed in Section 4.3. The end of this section discusses the fine points of the architecture (Section 4.4) as well as the resulting API (Section 4.5), while in Section 5 we show how the new architecture also solves the security problems currently hampering mobility and multihoming.

4.1. Layered structure

It is easiest to describe our new architecture by comparing it to the existing one. Figure 5 describes the current architecture. In that, processes are bound to transport layer sockets, and the sockets are identified by using IP addresses and ports. More formally, the ports may be called *Transport Layer Identifiers (TLI)*. As a result, this structure binds the processes to a specific topological location, thereby making process migration, end-host mobility, and multi-homing hard.

The new structure is described in Figure 6. In the new architecture, the transport layer sockets are no longer named with IP addresses but with separate *host identifiers*. The host identity layer translates the host identifiers into IP addresses. This is achieved by *binding* a Host Identifier to one or more IP addresses. This binding may be a temporally dynamic relationship, resulting in mobility support,

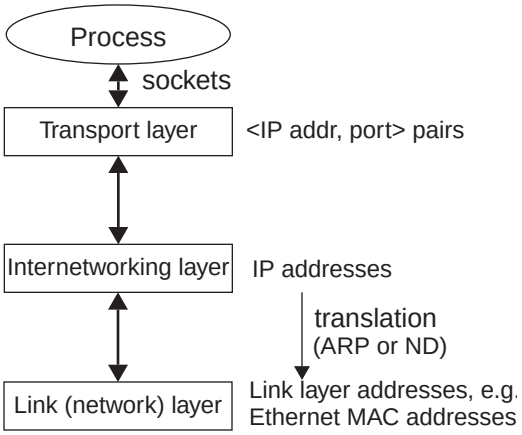

Figure 5: The current architecture

and simultaneously a one-to-many relationship, providing multi-homing support.

Bindings. Compared to the current architecture, the new architecture results in different bindings between the entities and identifiers. This is illustrated in Figure 7. In the current architecture, IP addresses are used to denote both hosts (end-points) and topological locations. In the new architecture, these functions have been separated, and the hosts (end-points) are denoted with Host Identifiers. Furthermore, the binding between a Host Identifier and the IP address(es) is made dynamic. As we explain in Section 5, due to the cryptographic nature of the Host Identifiers, it is fairly easy to secure the signalling messages needed to update this binding.

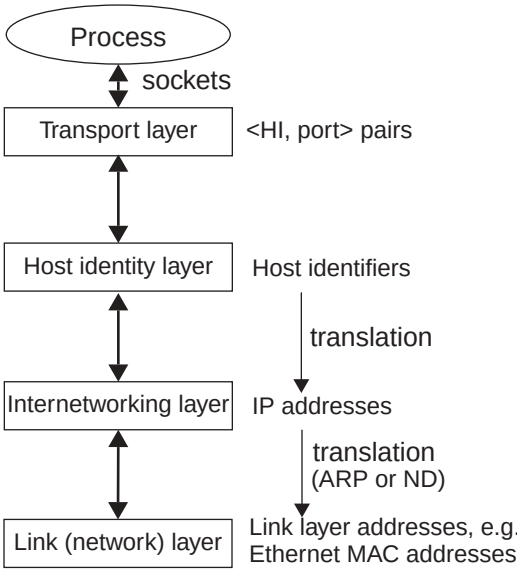

Figure 6: The proposed new architecture

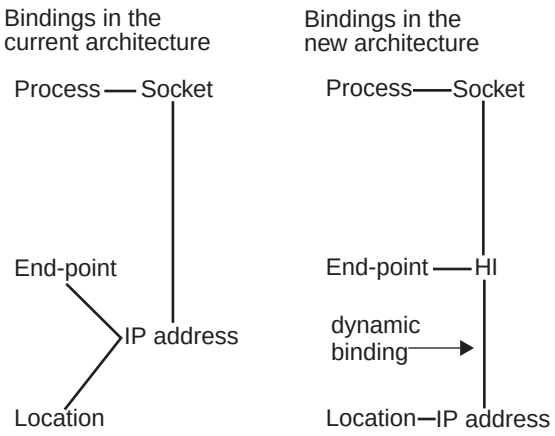

Figure 7: Bindings

Packet structure. At the logical level, the new architecture also requires changes to the packet structure. That is, each packet must logically include the Host Identifiers of the sender and recipient. However, whenever IPsec is used, the IPsec Security Associations can be used as a short-cut for Host Identifiers, resulting in packets that are similar to those used today. This is illustrated in Figure 8, and further discussed in Section 5.2

4.2. Components in detail

To make the architecture definition both definite and rooted to reality, we next precisely define the relevant concepts and terminology. The relationships between the concepts are also described pictorially in Figure 9.

Interface. A network interface. Usually a network interface is a physical piece of equipment that a host uses to connect to a network. For example, an Ethernet NIC is such a piece of equipment. However, an interface may also be completely virtual.

Each interface can be assigned one or more addresses. The address(es) depend on the *location(s)* of the interface. Even though the actual assignment mechanism is irrele-

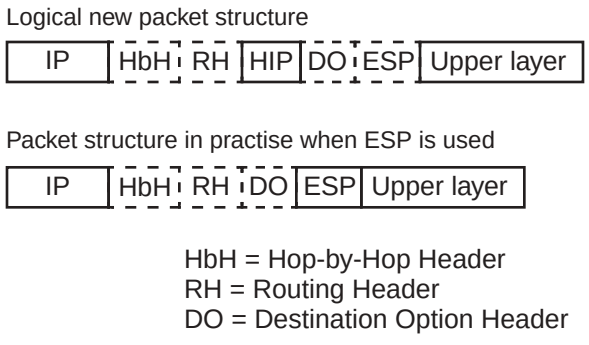

Figure 8: The packet structures

vant, it is important to understand that the assignment is either *determined* or at least heavily *influenced* by its topological point-of-attachment, i.e. location.

End-point. The logical end-point of communication, i.e. a participant in an end-to-end communication. [2][15][25]. In most cases end-points are identical to hosts, and it is usually safe to think about hosts when reading the architecture description. That is, typically a physical node hosts a single end-point.

Process. A communicating process. Usually an end-point hosts a number of processes. Sometimes an end-point hosts only one process, but even then the end-point and the process should be conceptually separated. Within an end-point, the processes are distinguished with Transport Level Identifiers (TLI), e.g. TCP and UDP ports.

Location. A topological point-of-attachment at the network. An end-point is said to be reachable at a certain location if packets sent to that location are delivered to the end-point. In [2] and [14] these are called (Network) Attachment Points.

Each location is assigned an address by the network. We consider these addresses static, or at least very slowly changing. In the case a single network provides several global addresses for each host attached to the network (site multi-homing), we consider that particular network to represent several topological locations, one location per address.

Address. A name of a location. In addition to acting as location names, addresses also function as (partial) routing selectors. That is, the routers within the internetwork use the address (and possibly other data) in making the decision where a packet is passed next.

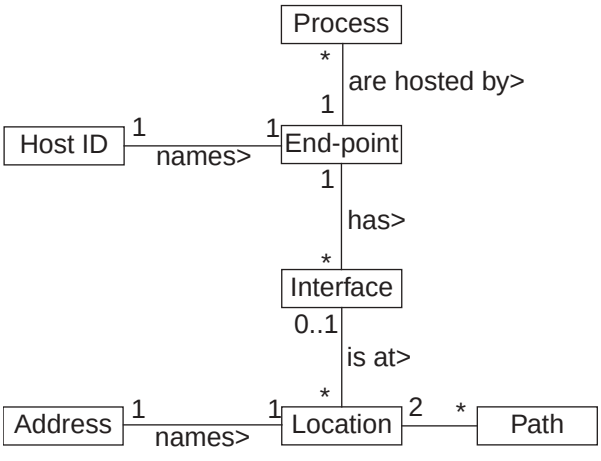

Figure 9: The conceptual model, in UML

Topological path. A path, through the internetwork, from one location to another location. We only consider those paths separate that can be distinguished on a level *above* the routing infrastructure. That is, parallel links and redundant routes appearing within the routing function are not considered separate topological paths. A topological path can be named with an address pair.

Multi-homed end-point. An end-point that is *simultaneously* reachable at more than one location. Usually this is a result of the end-point having multiple interfaces, each separately connected to different locations in the network. In the case of site multihoming, however, the whole site appears at two (or more) topologically distinct locations. In this latter case, the end-point may have just one interface, but that interface is considered to be simultaneously at more than one location, and therefore assigned more than one address.

Mobile end-point. An end-point that is *serially* reachable at more than one location. Usually this is a result of an end-point changing the location of (one of) its interface(s). In a sense, mobility is the dual of multi-homing in the same sense serialism is the dual of parallelism.

Host Identifier. A public key of a key pair, used to identify an end-point. We use the term Host Identifier (HI) instead of the more accurate term end-point identifier, mainly because we rely quite heavily on the HLP/HIP proposal and want to be consistent with its terminology.

Each physical node is assumed to generate one or more public key pairs. The public key of such a pair is used to identify an end-point hosted on that node. For the purposes of this paper, it is safe to think that each host is uniquely identified with a single key pair, and therefore is identical to a single end-point. However, there are reasons (such as anonymity, see [26]) to allow a host to represent itself as a set of end-points, and to allow the end-points to move between hosts; see [1].

4.3. Mobility and multihoming

It should be obvious by now that basic mobility and multihoming becomes trivial in the new architecture. That is, to support mobility all that is needed is to make sure that the binding between (an interface belonging to) a Host ID and IP address(es) is dynamic. Respectively, to support multihoming all that is required is to make the binding into a one-to-many relationship. In practice, dynamism and multiplicity are achieved with signalling, see Section 5.2.

To be more specific, in the presented architecture the Host Identifiers are used to identify the communication end-points, and they have no permanent relationship with locations or IP addresses. IP addresses, on the other hand,

are used to identify only the topological locations, not end-points. Thus, as a result, the *actual* addresses used in a packet don't matter so much as they do in the current architecture, since the end-points are not identified with them. All that is required is that the end-points are able to determine the addresses currently used by their active peers.

Furthermore, if the packets are integrity protected with e.g. ESP or AH, the recipient is always able to verify that a received packet was sent by the alleged peer no matter what the source and destination addresses are. Thus, by binding IPsec Security Associations to Host Identifiers instead of IP addresses, the destination address becomes pure routing information, and the source address becomes almost obsolete [24]. Only during connection setup, when the hosts haven't authenticated each other, does the source address pay substantial role. Once the peer hosts have secure bindings between the HIs and addresses, the source address is not needed any more by the hosts, and its only function becomes to carry information about the topological path the packet has taken [24].

Packet forwarding agents. In the architecture, basic mobility support requires that the moving end-point sends signalling messages (location updates) to its peers. These inform the peer about the changes in the addresses that it can use to reach the host. Thus, in the basic case explicit packet forwarding is not needed, since the hosts are able to send packets directly to each other. However, this leaves two problems unaddressed. Firstly, there must be a mechanism that allows an end-point to be contacted independent of its current location. Secondly, if two end-points move at the same time, it is possible that the signalling messages cross each other and never reach their intended destination. This is usually called the double-jump problem [18]. Introducing packet forwarding agents allows us to solve these two problems. Thus, we define a *packet forwarding agent* as a network node that forwards all packets sent to a given IP address (virtual address) to another IP address (real address).

We now generalize the concept of packet forwarding agents and, at the same time, fold them into our architecture. As discussed before, a multi-homed host is considered to be present at several locations at the same time. In functional terms, that means that the host is able to receive packets sent to several different IP addresses. On the other hand, if we have a packet forwarding agent, the end-host is also able to receive packets sent to the forwarded address. Thus, in a sense the *packet forwarding agent can be considered to represent a virtual interface* of the end-point, and that the *end-point is virtually present at the location of the forwarding agent.*

Thus, we define that *the location of an end-point is the topological point through which the end-point is able to receive packets.* It may be the location of a physical interface

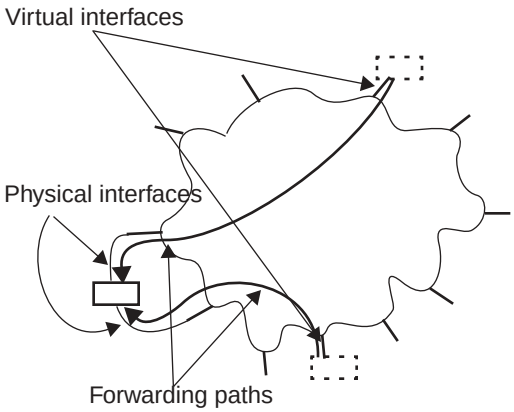

Figure 10: The virtual interface model

of the end-point, or it may be the location served by a packet forwarding agent. In the latter case, the packet forwarding agent is considered to act as a *virtual interface* for the end-point. The situation is illustrated in Figure 10.

4.4. Architectural elements

Now we are ready to define the functional components of our architecture. The basic components are the *internetwork*, the communicating *end-points*, and the (temporary) *packet forwarding* agents. Additionally, we need two external services and four protocols. Firstly, there must be a service and corresponding protocols that allow an end-point to learn the current set of addresses that another end-point has, i.e., an *address discovery* service. Secondly, a protocol is needed to allow end-points to inform their peers about *changes in the addresses* and status of their interfaces. Finally, a protocol is needed for creating new *packet forwarding agents*, and to signal changes to them. The architecture is described pictorially in Figure 11.

Internetwork. The internetwork is based on stateless IP level routers just as today. No changes are needed in the network itself. All the currently used IP routers will continue to function without any changes. This allows the new

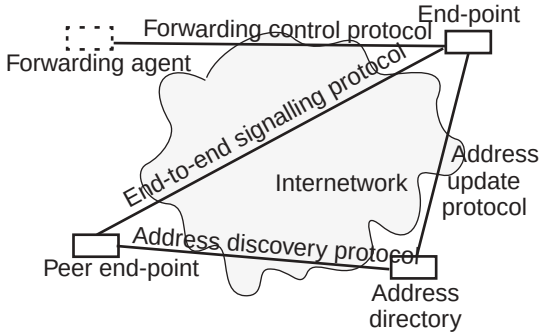

Figure 11: The elements of the architecture

architecture to be taken into use gradually, as the hosts adopt the new functionality.

End-points. The communicating end-points are hosted in network nodes.The end-points are able to communicate in two ways. Firstly, they may send plain IP packets just as today. In this case, the IP addresses are used to name the target and origin locations, and the end-point is supposed to stay at the same location long enough to receive replies. This form is suitable for fast low cost transactions, such as DNS queries. Secondly, the end-points may run the Host Layer Protocol (HLP), thereby authenticating the Host Identifiers of each other as explained in Section 5.2. This allows the end-points to communicate with added security, mobility, and reliability.

Packet forwarding agents. The packet forwarding agents are hosted within the internetwork; for example, in a typical case an access router would be willing to provide such an function (see Section 5.3). They allow, in a controlled way, end-points to receive packets that are sent to an address (virtual address) that the end-point does not currently control but that the forwarding agent does.

Address discovery. An initial contact between two end-points requires that the initiating end-point learns at least one IP address of the other end-point. This discovery function is supposed to be implemented by a directory service, such as the DNS. The details of such a service are beyond the scope of this paper.

End-to-end signalling. The updates in the end-point interface status must be signalled to the peer end-points. This is accomplished using the HIP Readdress Packets (REA) [7]. The REA packets are protected in the Host Identity Security Context; see Section 5.2 for details.

Forwarding control. The protocol to signal packet forwarding is a function not defined in the HIP specifications, and therefore we discuss the situation in Section 5.3.

4.5. Internal interfaces and APIs

The conceptual structure of a node hosting end-points is depicted in Figure 12. The node has a number of interfaces, both physical and virtual. Within the host, the network layer implementation takes care of routing, interface selection and other functions as today. The new host identity layer implements the new functionality. It is responsible for implementing the HLP/HIP end-to-end signalling protocol, and also the forwarding control and directory query and update protocols.

The transport layer implements transport protocols, e.g. TCP and UDP, like today. The only difference is that the transport layer sockets are bound to Host Identifiers instead

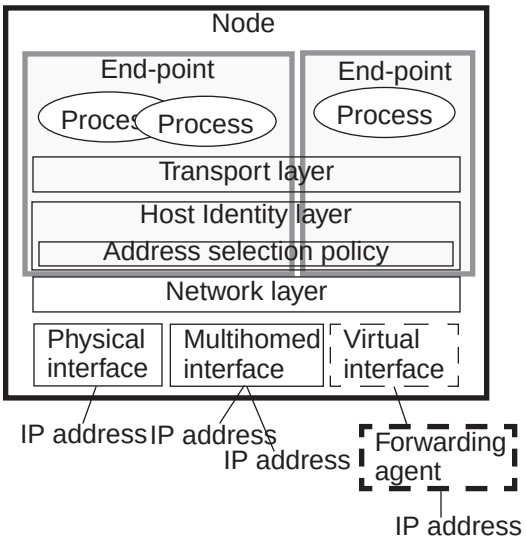

Figure 12: The structure of an end-point

of IP addresses. Finally, the communicating processes function as today. The only difference is that instead of IP addresses they use Host Identifiers. This is achieved in a completely backward compatible way, where all well written (IPv6) applications will continue to function without recompilation. The basic idea is to reserve a large fraction (actually half) of the IPv6 address space to represent host identifiers [8]. That is, the IPv6 compatible format would be that of an Host Identity Tag (HIT), which basically is the result of applying a hash function on the Host Identifier.

In an HIP aware host the DNS resolver library would return an HIT if one is available, and otherwise an IPv6 address. Transport protocols can then handle the HITs or IPv6 addresses transparently. In the case of HITs, the host identity layer would then perform the appropriate conversion to both incoming and outgoing packets in a way that is very similar to the so called host NAT [3].

5. Security

In the original TCP/IP architecture, a host's identity is implicitly authenticated by the routing infrastructure. That is, since the hosts are identified with IP addresses, and since IP addresses are the fundamental piece of data used in routing, the very definition of the internetwork assures that the IP packets are indeed sent to the intended hosts. (See also [24].) In the new architecture, there is no implicit binding between the host identifiers and the routing infrastructure. Thus, the implicit authentication does not exist any more, and must be replaced with an explicit one. Additionally, we must address the problems of address stealing and address flooding that were described in Section 2.3.

The address stealing and address flooding problems are not introduced with the end-point concept or the new host identifiers. Instead, they originate from the dynamic binding between the hosts themselves and their IP addresses. Thus, they exist already in environments that use dynamic IP address assignment: the address stealing and flooding problems are present even in plain vanilla Mobile IP. Fortunately, introducing public key cryptography based host identifiers that *are* public keys makes it easier to address these problems. In this section we look at the situation more closely, starting from the nature of the new identifiers, and continuing to the properties of the new signalling protocols and new functions.

5.1. Host Identifiers

The cryptographic nature of the Host Identifiers is the security cornerstone of the new architecture. Each end-point generates exactly one public key pair. The public key of the key pair functions as the Host Identifier. The end-point is supposed to keep the corresponding private key secret and not to disclose it to anybody. (Note, however, that e.g. due to privacy reasons a single *user* may want to be represented by several end-points at the network.)

The use of the public key as the name makes it possible to directly check that a party is actually entitled to use the name. A simple public key authentication protocol, such as the one included in the HIP exchange, is sufficient for that. Compared to solutions where names and cryptographic keys are separate, the key-oriented naming does not require any external infrastructure to authenticate identity. In other words, no explicit Public Key Infrastructure is needed. Since the identity is represented by the public key itself, and since any proper public key authentication protocol can be used to check that a party indeed possesses the private key corresponding to a public key, a proper authentication protocol suffices to verify that the peer indeed is entitled to the name.

This property of being able to verify the identity of any party without any explicit external infrastructure is the very cornerstone of our architecture. It allows the architecture to scale naturally, without requiring extra administrative overhead.

5.2. Host Layer Protocol (HLP)

The Host Layer Protocol/Host Identity Payload (HLP/HIP) is the end-point to end-point signalling protocol in our architecture. The details of the current protocol proposal are available as internet drafts [1][7][8] and beyond the scope of this paper. However, the security properties of the protocol are significant and explained briefly.

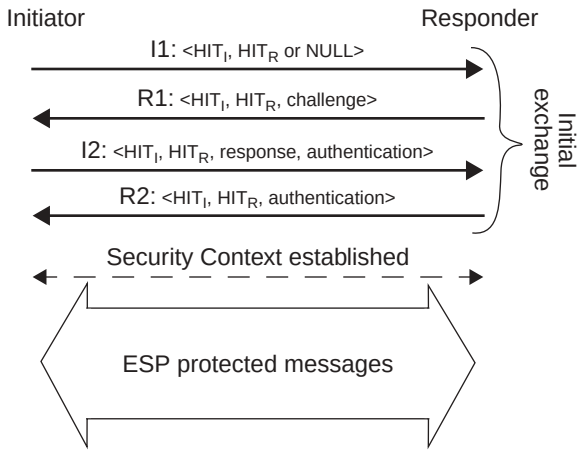

Figure 13: A typical HIP session

Most importantly, the HLP/HIP protocol performs mutual end-to-end authentication. This is accomplished with a four-way handshake, consisting of messages I1, R1, I2 and R2. After exchanging the initial HLP messages, both communicating hosts know that at the other end-point there indeed is an entity that possesses the private key that corresponds to its Host Identifier. Additionally, the exchange creates a pair of IPSEC Encapsulated Security Payload (ESP) security associations, one in each direction. The hosts are supposed to use the ESP security associations to protect the integrity of the packets flowing between them; optionally, ESP can also be used to encrypt the packets.

More formally, the initial HLP/HIP message exchange creates an Host Identity Security Context that contains the public keys of the communicating end-points, the ESP security associations, and implicit knowledge that the public keys were authenticated and present when the context was created. The flow of a typical HIP session is illustrated in Figure 13. Note that in the first message (I1) the responder's HIT may be NULL, indicating *opportunistic* mode of operation.

In addition to protecting the network layer integrity of the payload traffic, the Host Identity Security Context is used to secure the signalling messages exchanged between the end-points. For example, once the initial messages are exchanged and the security context is in place, the end-points inform their peers about the interfaces they have and the current IP addresses assigned to the interfaces. In effect, this shares information about the current multi-homing situation of the end-points. Each end-point has complete freedom to select which interfaces to announce to the peer.

To the peer, it is immaterial whether the announced interfaces are real or virtual. All it needs to know is to make sure that the end-point is indeed *reachable* through the claimed IP addresses. The reachability needs to be checked, or otherwise the mechanism may be used to

launch a number of complicated Denial-of-Service attacks. At the IETF mobile-ip Working Group, the reachability verification requirement has been extensively studied, and recently named Return Routability (RR).

Thus, initially all announced addresses should be considered as unverified. Using an RR protocol the peers can verify the reachability of the addresses given. Since there is already a pair of ESP Security Associations, the simplest way of implementing RR is to send an ESP protected challenge packet to the given address, and making sure that an ESP protected response is received. Once a challenge-response pair has been exchanged for a given address, the address can be marked functional.

5.3. Packet forwarding

From the security point of view, packet forwarding creates new security vulnerabilities. Firstly, packet forwarding could allow packets going to a legitimate host to be diverted either to an attacker or to a bogus location. For example, if an access router acts as a packet forwarder, this kind of attack could be used to create Man-in-the-Middle, masquerade, or Denial-of-Service situations against any host behind the access router. Secondly, packet forwarding could be used to divert existing packet streams to a host that is a target of a Distributed Denial-of-Service attack, see e.g. [11][27]. Thus, before starting to forward packets, the forwarding agent *must* make sure that there is nobody else but the requestor who wants to receive packets sent to the intercepted address, and that the requestor itself is able to receive packets sent to the new destination.

To generalize, we want to have an architecture where a mobile host may ask for a forwarding agent to be established (almost) anywhere in the network, and may establish a temporary virtual interface with the help of such a forwarding agent. However, from the security point of view, it is easier to ask a current or former Access Router to act as a packet forwarder than to request a generic node to provide packet forwarding service. We next look at these two situations.

Access Router as Forwarding Agent. If an end-point wants to use its current or recent Access Router as a packet forwarder, the case is security wise slightly easier than the generic case. As we discussed earlier, in our architecture we assume that the network assigns addresses (or at least routing prefixes) to the interfaces. Now, under these arrangements, the Access Router could easily know to whom it has assigned (or who has claimed ownership over) an address and for how long. Thus, if the end-point later, while the address assignment is still valid, requests the Access Router to start forwarding packets sent to that address, the Access Router can protect against address stealing just by checking that the requestor is indeed the same end-point that the address was assigned to.That is, the forwarding request needs to be signed by the end-point, and the access router simply checks that the signature matches with the address owner's public key.

Unfortunately the simple signature check does not completely protect against packet flooding denial-of-service, and therefore a Return Routability challenge-response is still needed.

Arbitrary node as Forwarding Agent. To support packet forwarding at an arbitrary location, e.g. for Local Mobility Management (LMM), one possibility is to devise *virtual access routers*. A virtual access router would be a router that serves addresses for forwarding purposes only. That is, any node in the network could ask for an address from it, and if given an address, ask packets sent to that address to be forwarded to another address. Thus, an end-point needing a virtual interface at a location served by such a virtual access router would contact the virtual access router for a new address. The access router assigns an unused address to the end-point, and the end-point requests that packets sent to that address are forwarded another address, the endpoints current real address. The access router then checks that the end-point is really reachable at the forward destination, and given so, starts forwarding packets.

5.4. Privacy

Using public keys as primary identifiers is clearly a potential source of privacy problems. If each user had just a single public key and that key is repeatedly used by the user, the very nature of public key cryptography leads to a situation where it is fairly easy to link together all the transactions made by the user. In the case of the HLP/HIP architecture, the situation does not need to be that bad. Since a single computer may host several end-points and therefore have several Host Identifiers, it is easy for a user to have several public keys instead of just one. One public key can be used as a more permanent identifier, allowing others to contact the user, while other keys can be completely temporary and periodically replaced with new ones. A temporary Host Identifier needs to be valid only as long as there are active connections associated with it.

5.5. Security summary

From the discussion so far it should be clear that addressing the mobility and multi-homing related security issues is much easier with the presence of cryptographic Host Identifiers than without them. Basically, we have two security issues that we have to address: firstly, we have to take care of basic identity authentication and related issues, and sec-

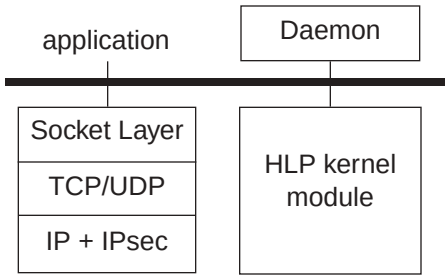

Figure 14: Implementation architecture

ondly we must take care of packet forwarding security. As an added benefit, we are also able to enhance privacy.

Since we are using public keys as the names for endpoints, identity authentication is trivial. All that is required is a good authentication and key agreement protocol; no certificates or external entities are needed. Basic multihoming and mobility brings in the requirement of checking each new IP address for reachability, to make sure that the host currently is reachable at the IP addresses it claims to be at.

Packet forwarding brings forth a couple of new security issues, and the node receiving a forwarding request must take care of these. Firstly, the forwarding agent either must know that the intercepted address really has been assigned to the requesting node and that the assignment is still valid, or it must know that the address is currently *completely* unassigned, and that it therefore can be securely assigned to the requesting node.

The second issue with packet forwarding is to make sure that the requestor is reachable at the address where the forwarded packets should be sent to. The easy way to take care of that is to use a challenge-response protocol to check that the requestor is reachable at the target address.

6. Implementation status

We have implemented HLP/HIP for NetBSD 1.6. The design consists of the daemon and loadable kernel module; see Figure 14. The kernel module registers hooks at the IP and socket layers. The core protocol state-machine and actual packet handling are implemented in the kernel while the multi-threaded daemon takes care of cryptographic calculations. Communication between the daemon and kernel is asynchronous.

To get an idea about performance, we measured the time taken by the four-way handshake, with different puzzle challenge values K [1]. The K is sent by the responder to the initiator, and a larger K value forces the initiator to make more work before the responder is ready to accept the connection. The idea is that the responder can easily check that the initiator has indeed performed the required

Table 1: Explanations for Figure 15

dT_1	Initiator generates I1
dT_2	Packet transit delay
dT_3	Responder generates R2t
dT_4	Packet transit delay
dT_5	Initiator processes R1, solves the puzzle, and generates I2
dT_6	Packet transit delay
dT_7	Responder processes I2 and generates R2
dT_8	Packet transit delay
dT_9	Initiator processes R2

work before performing the computationally expensive public key operations. Thus, if the responder is under a resource exhausting DoS attack, it can partially mitigate the effects of the attack by requiring legitimate initiators to make some work that allows it to cheaply distinquish legitimate initiators from DoS packets. The idea is that if the attacker would start making the work, it would slow down the attack since the attacker would need to spend a huge number of CPU cycles to pass the puzzle test.

Figure 15 illustrates the protocol run and the measured times; see also Table 1. In our current implementation the responder always generates R1 in response to a received I1. It is possible to move R1 generation into precomputation step, thereby reducing dT_3 by a large factor.

The measurements were made between two 800 MHz Pentium III PCs, running over a lightly loaded 100 Mbps switched Ethernet. The results are depicted in Tables 2 and 3. The figures consist of averages over 5 test runs ± one

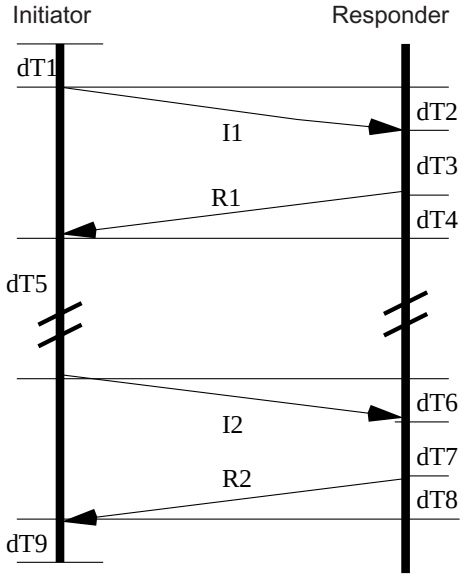

Figure 15: The times measured

Table 2: Average message processing time in ms

K	dT_1	dT_3	dT_5	dT_7	dT_9
0	1.4±0.3	115±9	300±12	176±10	26±3
7	1.4±0.3	121±12	503±99	177±10	34±6
8	1.1±0.0	117±8	573±250	185±16	35±5
9	1.4±0.3	134±6	2300±1370	158±3	30±7
10	1.2±0.0	129±5	1810±470	178±14	45±5

standard deviation. While this gives good estimates for the other time periods, the numbers for dT_5 are slightly misleading. That is, solving the puzzle is an indeterministic operation, requiring a random number of trials. Getting good numbers for dT_5 would require a large number of trials, in the order of thousands of runs.

As can be seen from the results, the total protocol run takes about 600 ms. About 500 ms of this is spent on the cryptographic operations. The introduction of a low puzzle factor K increases the average time by 200...300 ms; with K=10 the average protocol run takes still less than 3 seconds while requiring almost always over 1 second of CPU time for solving the puzzle. Our earlier tests indicate that when K > 10 the time required starts to grow fast, reaching 100 seconds with K = 15.

While the base HIP/HLP implementation seems simple enough, cleaning up and handling all the performance optimizations on the TCP side requires extensive modification to the kernel. Basically, it looks like many of the TCP algorithms must be modified to understand that the data may take different paths through the network, thereby complicating the statistics gathering and prediction algorithms. On the other hand, we expect that the SCTP implementors have already faced these problems, and our intention is to analyse the recently released kernel based SCTP implementation to see how the problems are addressed there.

7. Conclusions

The focus on this paper has been on modifying the TCP/IP architecture to include a new cryptographic name space and a new protocol layer. We have provided one possible design, heavily based on the HLP/HIP approach. We have also shown how the HLP/HIP approach can be easily extended to handle end-host multi-homing and mobility, and how to solve the involved security problems in a way that does not require any additional infrastructure. Furthermore, we have briefly touched the backward compatibility and API issues.

Table 3: Average packet delay in milliseconds

dT_2	dT_4	dT_6	dT_8
0.12	0.32	0.34	0.13

To sum up, the HLP/HIP approach provides new end-point names that *are* public keys. For convenience, the public keys are usually represented by tags derived by taking a cryptographic hash function over the key. The tags are used instead of IP addresses when representing the communicating parties to the applications. Along with the new names, a new layer is established between the network and transport layers. This layer takes care of establishing secure connection between any two end-points, translating the outgoing end-point names into IP addresses and determining the names from the security associations on incoming packets, and securely modifying the translation state to reflect the current multi-homing and mobility status. Additionally, since the communication context is bound to the end-point identifiers instead of IP addresses, the architecture also makes it easier to support several routing realms and to establish state with any node in the network. On the other hand, the security context is not, as such, suitable as a generic application level end-to-end security solution. To achieve application level semantics, the end-points need additional assurances about their peers.

From the architectural point of view, in our architecture it is sufficient to have just one mechanism to solve the reachability, double-jump, and local signalling optimization problems. From the security point of view, no separate mechanism is needed to secure mobility related signalling since the security inherent to the architecture suffices. In a companion paper [6] we have shown how the architecture can be extended to address the network mobility problem.

Acknowledgements

This work would have been impossible without the pioneering work performed by a number of senior researchers and engineers at the IETF, IRTF, and elsewhere. We are especially indebted to the insights of J. Noel Chiappa, and their application in the form of the HLP/HIP proposals by Robert Moskowitz. The insights made by the IRTF Name Space Research Group have allowed us to further refine our thinking.

We want to thank our colleagues Catharina Candolin, Miika Komu, Yki Kortesniemi, Glenn Morrow, Teemu Rinta-Aho, Göran Schultz, Vesa Torvinen, Zoltan Turanyi, and especially Juha Heinänen, Erik Nordmark, and Jarno Rajahalme, for their constructive comments on various versions of this paper. We also want to thank Petri Jokela and Tony Jokikyyny for their contributions to the actual text.

References

[1] R. Moskowitz, *Host Identity Payload Architecture*, work in progress, Internet Draft (expired), February 2001, http://homebase.htt-consult.com/draft-moskowitz-hip-arch-02.txt

[2] J. N. Chiappa, *Endpoints and Endpoint Names: A Proposed Enhancement to the Internet Architecture*, unpublished note available at http://users.exis.net/~jnc/tech/endpoints.txt

[3] S. Bellovin, *EIDs, IPsec and HostNAT*, a presentation give at 41st IETF in Los Angeles, California. Steven Bellovin, March 1998, http://www.research.att.com/~smb/talks/hostnat.pdf

[4] E. Lear, What's In A Name: Report from the Name Space Research Group, work in progress, Internet Draft draft-irtf-nsrg-report-02.txt, Internet Research Task Force, February 2002.

[5] E. Nordmark, *MIPv6: from hindsight to foresight?*, work in progress, Internet Draft draft-nordmark-mobileip-mipv6-hindsight-00.txt, IETF November 14, 2001.

[6] P. Nikander and J. Arkko, "Delegation of Signalling Rights," a position paper presented at the 10th Annual Workshop on Security Protocols, Cambridge, April 17–19, 2002.

[7] R. Moskowitz, *Host Identity Payload and Protocol*, work in progress, Internet Draft draft-moskowitz-hip-05.txt, November 2001, http://homebase.htt-consult.com/draft-moskowitz-hip-05.txt

[8] Robert Moskowitz, *Host Identity Protocol Implementation*, work in progress, Internet Draft (expired) draft-moskowitz-hip-impl-01.txt, Feb 2001, http://homebase.htt-consult.com/draft-moskowitz-hip-impl-01.txt

[9] G. Montenegro and C. Castelluccia, *SUVC Identifiers and Addresses*, work in progress, Internet Draft draft-montenegro-sucv-02.txt, November 2001.

[10] P. Bhagwat, C. Perkins and S. Tripathi, "Network Layer Mobility: an Architecture and Survey", *IEEE Personal Communications Magazine*, June 1996.

[11] A. Mankin et. al., *Threat Models introduced by Mobile IPv6 and Requirements for Security in Mobile IPv6*, work in progress, Internet Draft draft-ietf-mobileip-mipv6-scrty-reqts-02.txt, November 2001.

[12] P. Nikander, *Denial-of-Service, Address Ownership, and Early Authentication in the IPv6 World*, presented at Cambridge Security Protocols Workshop 2001, April 25-27, 2001, Cambridge University. To be published in the workshop proceedings at the LNCS series.

[13] G. O'Shea and M. Roe, *Child-proof Authentication for MIPv6 (CAM)*, ACM Computer Communications Review, Volume 31, Number 2, ISSN # 0146-4833, ACM April 2001.

[14] J. H. Saltzer, "On The Naming and Binding of Network Destinations," in *Local Computer Networks*, edited by P. Ravasio et al., North Holland, Amsterdam, 1982, pp. 311-317. Also available as RFC 1498, University of Southern California, Information Sciences Institute, Marina Del Rey, Calif., August 1993.

[15] J. H. Saltzer, David Reed and David Clark, "End-To-End Arguments in System Design", *ACM Transactions on Computer Systems*, Vol. 2, No. 4, November 1984.

[16] R. Stewart et al., "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", Internet Draft draft-ietf-tsvwg-addip-sctp-06.txt, work in progress, IETF, September 2002.

[17] N. Montavont, T. Noel, and M. Kassi-Lahlou, "MIPv6 for Multiple Interfaces", work in progress, Internet Draft draft-montavont-mobileip-mmi-00.txt, July 2002.

[18] C. Huitema, Multi-homed TCP, work in progress, Internet Draft (expired), May, 1995, http://www.chem.ucla.edu/~beichuan/etcp/huitema-TCP.txt

[19] F. Teraoka et. al., LIN6: A Solution to Mobility and Multi-Homing in IPv6, work in progress, Internet Draft draft-teraoka-ipng-lin6-01.txt, 16 August 2001.

[20] M. Crawford et. al., *Separating Identifiers and Locators in Addresses: An Analysis of the GSE Proposal for IPv6*, work in progress, Internet Draft (expired), draft-ietf-ipngwg-esd-analysis-05.txt, October 1999, http://www.ietf.org/proceedings/99nov/I-D/draft-ietf-ipngwg-esd-analysis-05.txt

[21] P. Nikander, C. Candolin, and J. Lundberg, "From address orientation to host orientation," in *Réseaux et systèmes répartis, calculateurs parallèles*, ISSN 1260-3198, Special Issue on Mobility and Internet, Volume 13, Nr:o 2, Hermes Science Publications, Paris, France, December 2001.

[22] P. Nikander, "An Address Ownership Problem in IPv6," work in progress, Internet-Draft (expired), February 2001, http://www.tml.hut.fi/~pnr/publications/draft-nikander-ipng-address-ownership-00.txt

[23] A. C. Snoeren and H. Balakrishnan, "An End-to-End Approach to Host Mobility", *Proc. of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, August 2000.

[24] C. Candolin and P. Nikander, "IPv6 Source Addresses Considered Harmful," in Hanne Riis Nielson (ed.), *Proceedings of NordSec 2001, Sixth Nordic Workshop on Secure IT Systems*, November 1-2, Lyngby, Denmark, Technical Report IMM-TR-2001-14, pp. 54-68, Technical University of Denmark, November 2001.

[25] B. Carpenter, "Architectural Principles of the Internet", RFC 1958, IETF June 1996.

[26] R. Moskowitz, *The Need for a new Internet Namespace*, informal note in circulation, Robert Moskowitz, November 1999.

[27] T. Aura and J. Arkko, "MIPv6 BU Attacks and Defences", work in progress, Internet Draft draft-aura-mipv6-bu-attacks-00.txt, November 2001.