# A fresh look at IP based Mobility and Multi-homing

Pekka Nikander

*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 made IP mobility and multi-homing unnecessarily hard from the security point of view.

In this paper we discuss how the Host Identity Payload (HIP), being considered 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.

## 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 Point-to-Point Protocol (PPP) and by Dynamic Host Configuration Protocol (DHCP), the assumption that an IP address would uniquely identify a host was broken. The situation was further worsened by the introduction of private IP address spaces and Network Address Translation (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[1].

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, there are new requirements for mobility and multi-homing together with the necessary host-to-host signalling security. 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.

### 1.1 Need for a new protocol layer

The biggest architectural re-engineering need seems to be the demand for a new name space and an associated protocol layer. As suggested, for example, by Chiappa [2] and Bellovin [3], there seems to be a strong need for adding new kinds of identifiers, end-point identifiers, somewhere between the network and transport layers. More recently, the Name Space Research Group (NSRG) of the Internet Research Task Force (IRTF) have also published their first interim report [4]; in the report the need for a new name space and protocol layer is extensively discussed, both pro and cons, but no clear conclusions are provided. The main argument against the new name space and layer seems to be the fear of introducing more complexity without solving any explicit

1 Sometimes such an environment is described with the term *simultaneous multi-access*, emphasizing that the end-host will have several parallel access methods in its disposal.

problem. Even though we acknowledge the value of this argument, we also want to note that the functional complexity inherent in the introduction of the new layer actually brings forth architectectural simplicity; see Section 5.

**Our position.** We believe that both a new name space and a new protocol layer are needed. A clear indication of this need is the sheer number of smallish protocols that people at the IETF are trying to squeeze in to the narrow space between the IP and the upper layers. For example, in addition to IPSEC and Mobile IP, we now have Network Address Translation (NAT) and even NAT related signalling protocols such as RSIP and MIDCOM, quality-of-service signalling mechanisms , and proposals for end-host multi-homing. Even the existense of IPv6 extension header mechanism itself is an acknowledgement of this need.

Chiappa ([2], Section 8.2) argues that the introduction of end-point identifiers does not *necessitate* a new layer. He writes that "layers should only be introduced when needed, and there does not seem to be a major utility in being able to use the internetwork layer without dealing with endpoints". We do think, as we mentioned above, that a new layer is needed indeed. Firstly, it seems that the required new functionality falls in two different categories: those functions that *do* depend on network topology and the specific location of the end-host (e.g. QoS) and those that *do not* (e.g. end-to-end security). Mobility and multi-homing fall in between, and seem to create a natural protocol boundary so that location dependent functions fall below it and location independent functions ascend above it (see Section 5.1).

Secondly, the added functionality provided by the new cryptographic identifiers imposes a performance cost in the form of an indispensable cryptographic protocol (see Section 4). Thus, the utility value Chiappa is calling for is created by the ability to select between added functionality and lower cost. A new layer can be used or bypassed, depending whether it is worth paying the cost for the added security and mobility support. Without a new layer such a choice would be hard to make. A new layer adds orthogonality by allowing this choice.

## 1.2 Original contributions

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*[2] are neatly separated into different dimensions.

One way of characterizing our approach is to compare it to current development in the mobile-ip and multi6 working groups at the IETF. At the mobile-ip working group, the architecture has been generally ready for a long time, but solving the associated security problems have taken a long time and produced a sub-optiomal result. At the multi6 working group, people have proposed several different host based approaches to solve the multihoming problem. Our approach, in a way, resembles the mobile IPv6 and host based multi6 approaches, but at the same time it provides a trivial solution for the mobility related security problems and many of the multi-homing related scalability problems. On the other hand, due to the architecturally different nature, our approach leads to a cleaner concepts and simpler implementation.

## 1.3 Paper structure

The rest of this paper is organized as follows. In Section 2 we discuss the nature of mobility and multihoming, thereby paving the way for the forthcoming discussion. Section 3 defines the proposed new architecture in detail, and Section 4 discusses it from the security point of view. In Section 5 we again analyze some of the issues already briefly mentioned in Section 1 and elsewhere in this paper. Finally, Section 6 concludes this paper.

# 2 Background

In this section we summarize the necessary background for the following discussion. We assume that the reader is familiar with the architecture of IPv4 and IPv6 based TCP/IP protocol suites, and knows the fundamental principles behind layered protocol design. Thus, we limit this background discussion to analyzing mobility and multihoming from an end-host point of view. Later, in Section 3, 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 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.

Table 1: Abbreviations used in the paper, in alphabethical order

| Abbrev. | Explanation |
|---|---|
| AH | Authentication Header.<br>An optional IP header used to protect the integrity and authenticity of IP packets. |
| DHCP | Dynamic Host Configuration Protocol.<br>A protocol used to assign an IP address and other information to hosts by a server. |
| ESP | Encapsulated Security Payload.<br>An optional IP header used to protect the integrity, confidentiality and authenticity of IP packets. |
| GPRS | GSM Packet Radio System.<br>An extension to the GSM wireless phone network to carry packet data. |
| HI | Host Identifier.<br>An end-point identifier proposed by the HLP/HIP architecture. |
| HIP | Host Identity Payload.<br>A proposal by Robert Moskowitz and others to add a new layer above the IP layer but below the transport layer. |
| HIT | Host Identity Tag.<br>An 128 bit representation of a Host Identifier, generated by taking a cryptographic hash of the HI. |
| HLP | Host Layer Protocol.<br>An alternative name for HIP, emphasizing the actual protocl used. |
| IPSEC | IP Security.<br>A suite of protocols used to secure IP packets. See also AH and ESP. |
| IRTF | Internet Research Task Force.<br>The research branch of the IETF. |
| LMM | Local Mobility Management.<br>A concept for hiding host mobility by managing it locally. See also MAP. |
| MANET | Mobile Ad Hoc Network.<br>An IETF WG studying networks consisting of independent mobile nodes. |
| MAP | Mobile Anchor Point.<br>A network node used in LMM. *LMM agent* in an alternative name for MAP. |
| MIDCOM | Middlebox communication.<br>An IETF WG studying non-router nodes in the middle of the network. See NAT. |
| NAT | Network Address Translation.<br>A practise of the network changing IP addresses on the fly |

| Abbrev. | Explanation |
|---|---|
| NSRG | Name Space Reseach Group.<br>An IRTF research group. |
| PFKEY | An IPsec API that allows a key management daemon to manage the cryptographic keys used by IPsec AH and ESP. |
| PPP | Point-to-Point Protocol. |
| QoS | Quality of Service. |
| REA | Readdressing packet.<br>Used in HIP to implement mobility. |
| RR | Return Routability.<br>A security mechanism used in MIPv6. |
| RSIP | Realm Specific IP.<br>An alternative for NAT. |
| SCTP | Stream Control Transport Protocol.<br>An new transport layer protocol, somewhat similar to TCP and UDP |
| TLI | Transport Layer Identifier. |
| WLAN | Wireless LAN |

## 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]). When discussing mobility, it is often desirable to differentiate between *user mobility, code or application mobility* and *node* or *end-host mobility.* Recently it has become apparent that this list needs to be augmented with *network mobility*, which refers to a situation where a whole subnetwork moves from one location to another. However, network mobility is beyond the scope of this paper, and treated separately in [5].

User mobility denotes functionality that allows users to move from one end-node to another and continue their tasks. In the extreme form it requires full process migration together with communication session migration, combined with adaptive user interfaces that allow the migrated process to adapt to the new execution environment. Code mobility, on the other hand, refers to functionality that is needed to support mobile agents and migrating processes. In a limited form, it just allows an existing process to be migrated to a new node, and some other mechanism is needed to re-establish the communications context.

Node mobility denotes functionality that allows a communications node to change its topological location in a network. In a typical case, a wireless node changes the access point it uses to communicate with a fixed network.

As we will see in Section 5.4, given the proper node mobility architecture, certain aspects of user and code mobility become easier. That is, once the architecture makes it easy for end-hosts to move, that is, to logical communication end-points to change their topological location, it also makes process migration easier.

**End-host mobility.** In this paper we concentrate on end-host mobility. 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. In other words, we want that the communicating processes can continue as unaffected as possible. In particular, we aim for an architecture where 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 (see e.g. [6]). 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.

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 signaling must be secure since unsecured signalling can lead to a unauthorized traffic diversion and denial-of-service attacks.

To really understand why the current situation makes end-host mobility unnecessarily hard, we have to consider the structure of the current IP architecture. Today there are exactly two name spaces that are related to mobility. Firstly, we have network layer *addresses*. As stated above, these addresses are determined by the network topology. Secondly, we have *Transport Layer Identifiers* (*TLIs, e.g.,* ports in TCP and UDP). Of these, the network addresses usually have a global scope[3], and the TLIs are unique within the scope of a single address. Since we have only these two name spaces, the communicating processes must be named with <address, TLI> pairs, binding the names effectively to the topological locations in the network. This situation makes it difficult to give unique names to processes that are mobile, e.g. hosted on a mobile node. That is, since the addresses

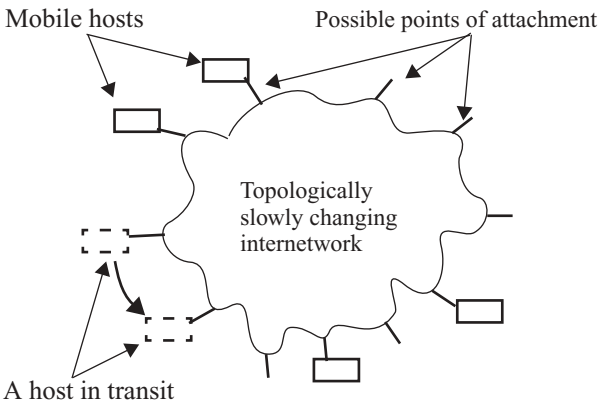


Figure 1: The mobility model

3 Or are translated into global scope addresses, if NAT is used.

must change due to mobility, the names of the end-points also must change. This is sometimes called the mobility problem [7].

There are the two fundamental approaches to solving the mobility problem: packet forwarding and dynamic updates of end-point bindings. *Packet forwarding* suffers from intrinsic performance penalty: the hosts cannot use optimal routes since they do not know the topological locations of their peers. *Dynamic updating of bindings* suffers from a number of security problems, as well as some other smaller problems [7].

In the dynamic update side, there are two basic reasons behind the security problems. Firstly, unsecured binding updates would allow various man-in-the-middle, masquerade, and denial-of-service attacks (see e.g. [7], [8]). Thus, one must employ a way to somehow secure them. This, in turn, requires creating authorization relationships between the parties involved. Secondly, even though in theory it would be possible to create a global authorization infrastructure that would allow the update messages to be secured, such an infrastructure does not exist, and creating one would be extremely difficult or impossible in practice. As we will see, the introduction of cryptographically assigned Host Identifiers changes the situation so that a replacement for such an infrastructure emerges almost "magically".

**Double jump.** If dynamic update is used, a situation where both communicating nodes move simultaneously becomes problematic. If there are no special arrangements, the updates will cross within the network and never reach the recipient nodes. Since both nodes have moved, they do not know the new addresses of their peers. Consequently, they cannot inform their peers about their new addresses, leading to communication outage. This problem is often called the *double jump* problem.

There are various approaches to solve the double jump problem. One is to combine packet forwarding and dynamic update of bindings. Another one is to rely on a separate naming service as the last resolt. That is, in the case of a double jump the nodes make a query to the naming service and thereby learn the new address(es) of their peers.

**Terminology.** In the rest of this paper we mostly follow the terms defined in the Mobile IPv6 standarization process. A *mobile node (MN)* is a mobile device hosting applications, i.e. an end-host that changes its point of attachment. For the purposes of this paper, a *mobile host* is synonym for a mobile node. A *corresponding node* is a peer of a mobile node, i.e. a host that communicates with a mobile host. A corresponding node may be itself mobile. A *home agent* is a stationary node that provides services to one or more mobile nodes. Usually the services include packet forwarding, but they may include also other services.

## 2.2 Multihoming

Multi-homing refers to a situation where an end-point has several parallel communication paths that it can use. Sometimes this is called multi-path or multi-access; in this paper we do not make a distinction between these, and just use the term multi-homing. 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). At the first look, it seems necessary to make a difference between site multihoming and end-host multihoming, since they seem to have different problems. For example, a fundamental problem associated with site multihoming is routing table management. That is, the simplistic approach of assigning a single network address prefix to the site and announcing this prefix through all paths leads to the so called routing table explosion, i.e., scalability problems.

End-host multihoming seems to be easier to solve. Today most multihomed end-hosts are servers, but, for example, a future mobile terminal with both GPRS and WLAN connections would also be a multihomed end-host. In this paper we mainly consider multihoming from the end-host point of view. However, in Section 5.2 we argue that the proposed architecture will also solve certain aspects of the site multihoming problem. Thus, maybe end-host multi-homing and site multi-homing are not that different, after all. In this paper we concentrate on end-host multihoming.

**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.

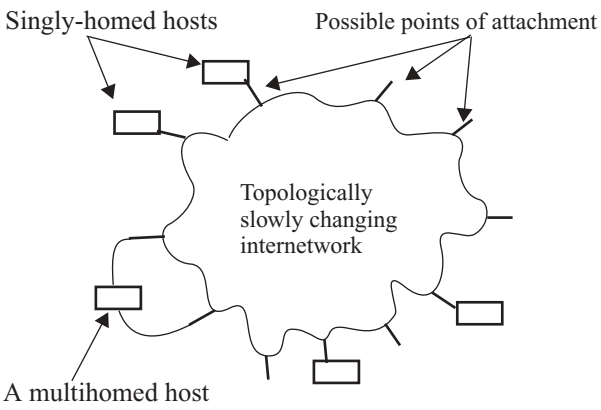


Figure 2: The multi-homing model

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.

**Connection management.** Having several independent network addresses makes the life of multihomed end-hosts hard. That is, since the communication processes are named with <address, TLI> pairs, the multihomed host must select which address to use when a connection is first opened. If it later wants to change the address used, the situation is identical to the identifier update situation with mobile hosts, and leads to exactly same security problems. Thus, as we will see, solving the mobility problem automatically solves the end-host multihoming problem as well.

## 3 Architecture

This section defines our Host Identifier based multihoming, mobility, and security architecture. In Section 3.1, we outline the overall architecture as a layered structure. In Section 3.2 we give the exact definitions for the required terminology and components. As it turns out, just defining the terminiology in a new way naturally leads to looking at the mobility and multihoming 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 3.3. The end of this section discusses the fine points of the architecture (Section 3.4) as well as the resulting API (Section 3.5), while in Section 4 we show how the new architecture also solves the security problems currently hampering mobility and multihoming.

### 3.1 Layered structure

It is easiest to describe our new architecture by comparing it to the existing one. Figure 3 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.

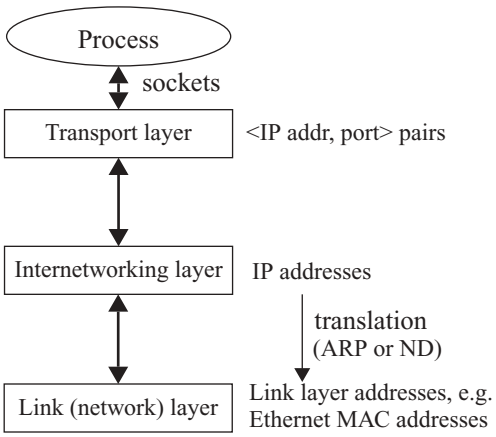


Figure 3: The current internetworking architecture

The new structure is described in Figure 4. 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, 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 5. 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

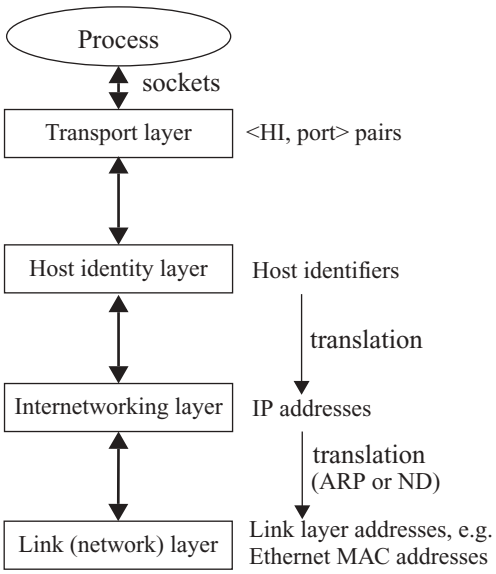


Figure 4: The proposed new architecture

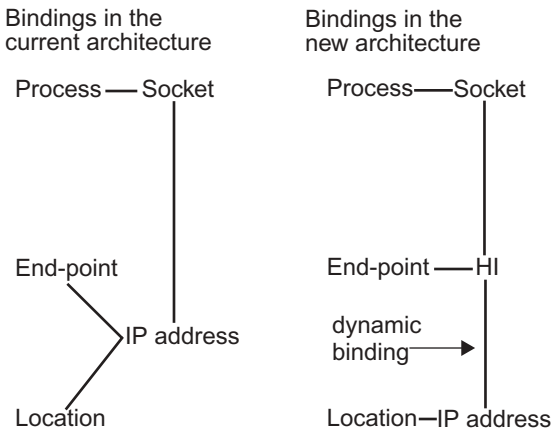


Figure 5: Bindings

Identifiers. Furthermore, the binding between a Host Identifier and the IP address(es) is made dynamic. As we explain in Section 4, due to the cryptographic nature of the Host Identifiers, it is fairly easy to secure the signalling messages needed to update this binding.

**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 shortcut for Host Identifiers, resulting in packets that are similar to those used today. This is illustrated in Figure 6.

## 3.2 Components in detail

To make the architecture definition both definite and rooted to reality, we next precisely define the relevant concepts and terminology. While most of the concepts are familiar to the intended reader, there are some subtleties that are important. The relationships between the concepts are also described pictorially in Figure 7.

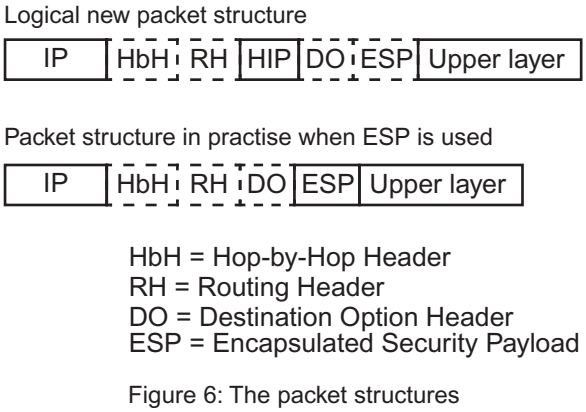


Figure 6: The packet structures

**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. For example, tunnel end-points and virtual machine interfaces are virtual interfaces.

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 irrelevant, 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 (see below).

**End-point.** The logical end-point of communication, i.e. a participant in an end-to-end communication. [2][9][10]. 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. The alternatives and fine details are discussed in Section 5.4. (See also Section 6 of [2].)

**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 [11] these are called as (Network) Attachment Points.

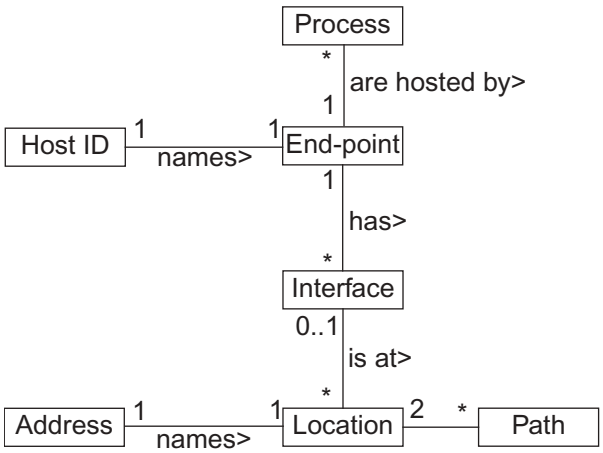


Figure 7: The conceptual model, expressed in UML

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.

**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, see Figure 8. 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, see Figure 9. 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.

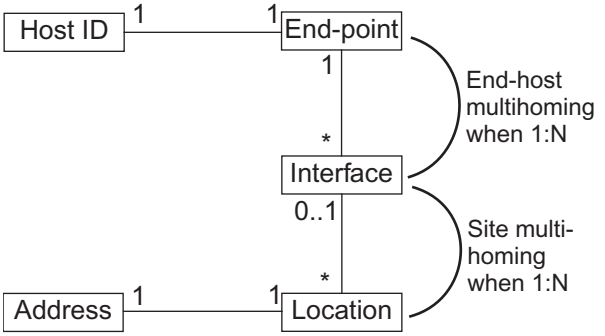


Figure 8: Two types of multihoming

**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 [12]) to allow a host to represent itself as a set of end-points, and to allow the end-points to move between hosts; see Section 5.4 and [1].)

The fact that the nodes generate their key pairs themselves lead to a suspicion that there is a possibility of key collision, i.e. two nodes generating the same key pair. However, the chance is extremely small. According to the Prime Number Theorem the density of primes near a given number $n$ is about $1/\log n$. Thus, using 1024 bit RSA keys, and correspondingly about 512 bit primes, the density of primes around $2^{512}$ is in the order of $1/512 = 1/2^9$, meaning that there are about $2^{(512-9)}$ primes of that size. According to the birthday paradox, in a population of approximately $\sqrt{2n}$ is expected to produce about one collision. Since the number of primes is about $2^{503}$, the size of a population required to produce an almost sure collision is about $\sqrt{2 \cdot 2^{503}} = \sqrt{2^{504}} = 2^{252}$. The other way of expressing the density is to say that if distributed evenly, there are more than $10^{140}$ 512 bit primes for each human.

## 3.3 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

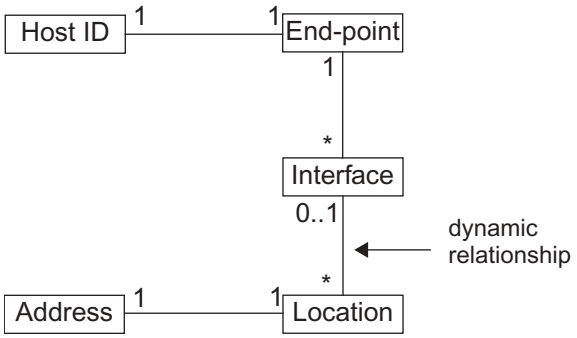


Figure 9: Mobility

sure that the binding between (an interface belonging to) a Host ID and IP address(es) is dynamic. Respectively, to support multi-homing all that is required is to make the binding into a one-to-many relationship.

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 IPSec, the recipient is always able to verify that a received packet was sent by the alledged peer no matter what the source and destionation addresses are. Thus, by binding IPsec Security Associations to Host Identifiers instead of IP addresses, the destination address becomes pure routinting information, and the source address becomes almost obsolete [13]. 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 [13].

However, beyond basic mobility and multi-homing achieved with the separation of the identifying functions, there are additional related problems that require the architecture to be slightly augmented. That is, to fully support mobility and end-host multi-homing, the base architecture must be augmented with *packets forwarding agents*.

**Packet forwarding agents.** Let us look at explicit packet forwarding first. 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. Introducing packet forwarding agents allows us to solve these two problems. In addition to solving these two problems, the solution introduced allows us to optimize and localize signalling.

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).

In the current Mobile IP architecture, there are three different kinds of entities that perform explicit packet forwarding: the so called Home Agent, the Access Routers, functioning as temporary Home Agents, and the Hierarchical MIPv6 [14] Mobile Anchor Points (MAP). Each of these functions slightly differently, and needs a slightly different security solution. However, looking at the problem more closely, it becomes obvious that in all these cases the explicit packet forwarding function is just a single function, and there is little added value by separating between different types of nodes.

Thus, to solve both the initial contacting and the double-jump problems, an actually *better* solution is one where there is just *one* packet forwarding function instead of two different ones. To address double jump, some node at the old location *temporarily* forwards packets to the new location. Such a temporary forwarding agent can be created at any convenient point, it is not restricted to the Access Router. To address the node reachability problem, it is *sufficient* that the packet forwarding function is available at *some* globally known address or addresses, e.g. one(s) available at the DNS. This set of nodes need not to be fixed, it just needs to be stable enough to compensate the scalability problems associated with dynamic DNS updates. Indeed, we claim that for the huge majority of the forthcoming IPv6 nodes, the idea of having a single permanent address is completely unnecessary and even unnatural. Furthermore, those nodes that benefit from a permanent address are typically large servers, often wanting to have *several* permanent addresses instead of a single one, and not moving at all.

In summary, explicit packet forwarding is clearly needed. We now generalize the concept of packet forwarding agents and, at the same time, fold them into our architecture.

**Real and Virtual Locations.** 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.*

Consider the situation from a multi-homed host's peer's point of view. The peer only sees a number of addresses, and it knows that the multi-homed host is reachable through all of these addresses. From the peer's point of view there is no difference whether packets sent to a given address are forwarded or directly received by the host. All that matters is that the packets eventually reach the right destination end-point. Furthermore, since all of the address bindings have a life time[4], the fact that the forwarding binding is a temporary one doesn't make a difference between direct and forwarded addresses.

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 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[5]. The situation is illustrated in Figure 10.

**Optimizing signalling traffic.** In addition to the above mentioned basic reasons, there are also other reasons why packet forwarding may be useful. Possibly Local Mobility Management (LMM) [15] is the most important of these. The goal of LMM is to reduce signalling traffic between a mobile end-point and its peers by introducing one or more Mobility Anchor Points (MAP, also called LMM agents) between the mobile end-point and its path to the fixed network. If these Anchor Points

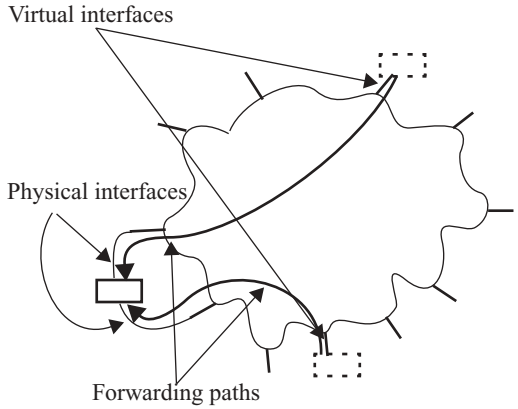


Figure 10: The virtual interface model

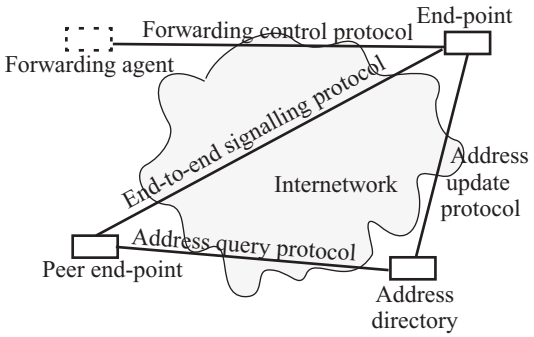


Figure 11: The elements of the architecture

function as virtual interfaces, the mobile end-point can hide the movement of its real interface from the peers and only announce the virtual one provided by the Anchor Point. Thus, whenever moving within the area served by the Anchor Point, it is sufficient that the mobile end-point signals the movement to the Anchor Point, and the peers may remain unaware of movement. This reduces the amount of signalling traffic required.

## 3.4 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 endpoint 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 connectivity 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 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. Each end-point is usually, but not nec-

4 Consider that in IPv6 basically all addresses have a lifetime. Therefore it is necessary to have lifetimes in the Host ID — IP address bindings as well. In the case of IPv4 addresses, the lifetime can be e.g. a few months.

5 The usual way of implementing packet forwarding is tunneling. Thus, denoting packet forwarding agents as virtual interfaces is not so different from considering tunnel end-points as virtual interfaces. However, in the host identity based architecture tunneling is not needed.

essarily, associated with a single physical node. 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 4.2. This allows the end-points to communicate with added security, mobility, and reliability. The internal structure of the end-points is discussed in Section 3.5.

**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 4.3). 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. That is, upon request and after the necessary security checks, a packet forwarding agent starts to intercept packets sent to a particular address and to forward them to another address. The first address is assumed to be one that is controlled by the forwarding agent; for example, such an address would be one that belongs to the subnet controlled by an access router functioning as a packet forwarding router. The second address is usually an address where the receiving end-point is actually located at, though it may be an address that is actually intercepted by another forwarding agent.

A forwarding agent will continue to intercept and forward packets only up to a maximum time, defined by a policy local to the agent. Typically, this time will be in the range of a few minutes, but in the case of semi-permanent agents it could be months or years. If desired, it is the responsibility of the end-point to renew the forwarding request before expiration.Thus, the forwarding state at the forwarding agents is based on the concept of leases, and therefore automatically cleared even in the case of errors.

**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, e.g. whether it is possible to use the Host Identifier as a key to look up IP addresses or some other type of a key is needed, are beyond the scope of this paper.

**Address updates.** The updates in the end-point interface status must be signaled to the peer end-points. This is accomplished using the HIP Readdress Packets (REA) [16]. The REA packets are protected in the Host Identity Security Context; see Section 4.2 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 4.3. The actual function is quite similar to the Mobile IPv6 Binding Update mechanism. That is, using Binding Update to signal a Home Agent to change its packet forwarding status is functionally similar to the packet forwarding signalling within our architecture. On the other hand, the security requirements and solution are somewhat different.

## 3.5 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. Usually each physical interface is singly homed and has one global IP address, and in IPv6 addionally one or more local addresses. However, the site itself may be multi-homed, in which case the host has several global IP addresses, and the same local addresses (if any) that it had before. The virtual interfaces represent forwarding agents and other tunnel end-points that can be used to send packets to the node. Each virtual interface is associated with one or more IP addresses.

Within the host, the network layer implementation takes care of routing, interface selection and other functions as today. It is also supposed to take care of network layer QoS services, if such exist. Basically, the network layer is very similar to the one in the current architecture.

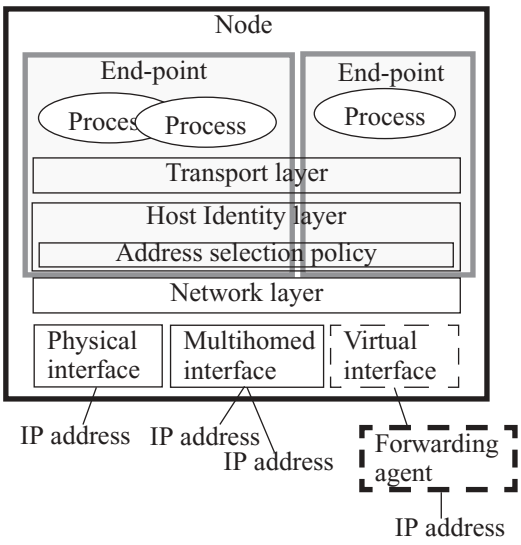


Figure 12: The conceptual structure of an end-point

The new host identity layer implements the new functionality. In a typical implementation, it would be located in the operating system kernel. In some cases it may provide facilities for hosting several parallel end-points, but more typically there will be just one end-point within a host. IIt is also responsible of implementing the HLP/HIP end-to-end signalling protocol, and at least conceptually also the forwarding control and directory query and update protocols. However, in an actual implementation large parts of these protocols would probably be implemented outside the operating system kernel, communicating with the host identity layer through administrative sockets similar to PFKEY or routing sockets.

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 of IP addresses. Technically this is achieved by representing the Host Identifiers in an IP address compatible way — see below. However, it is somewhat unclear how to deal with SCTP; the SCTP multi-homing functionality supports some of the multi-homing features in our architecture, but not all. For example, SCTP could even be used in a way where SCTP sees both a Host Identifier and a number of IP addresses. However, evaluating the benefits and drawbacks of different possibilities are beyond the scope of this paper.

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 faction (actually half) of the IPv6 address space to represent host identifiers [17]. 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[6].

In an HIP aware host the DNS resolver library would return an HIT if one is available, and otherwise an IPv6 address[7]. 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].

6 In this paper we do not consider old IPv4 applications at all. The HIP drafts propose the concept of a Local System Identifier (LSI) which could be used to make them to work as well, but that is beyound the scope of this paper.

# 4 Security

The introduction of host or end-point identifiers opens up new security vulnerabilities. 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 [13].) 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 a number of vulnerabilities emerge.

Fortunately, introducing public key cryptography based host identifiers that *are* public keys fixes almost automatically most of the new vulnerabilities. In this section we look at the situation more briefly, starting from the nature of the new identifiers, and continuing to the properties of the new signalling protocols and new functions. A more thorough security description is a subject of a companion paper [19].

## 4.1 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 suf-

7 If the peer has only an IPv4 address, the resolver library would return the IPv4 address in the IPv6 compatible format.

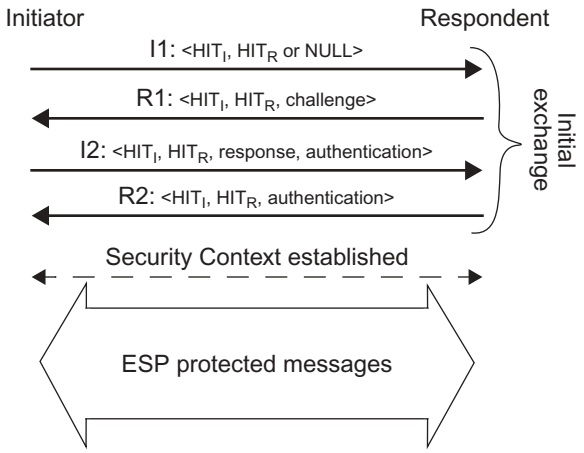


Figure 13: A typical HIP session

fices 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.

## 4.2 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][16][17] and beyond the scope of this paper. The basic security properties of the protocol are significant and explained briefly.

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 (see Figure 13). 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. Note that in the first message (I1) the responder's HIT may be NULL, indicating *opportunistic* mode of operation.

Once the authentication protocol has been exceuted and the initial multi-homing situation is established end verified, the end-points may communicate in an secure and resilient way. As the connectivity status of the end-points change they may signal the changes in the situation as needed. That is, if an end-host looses connectivity on an interface, acquires a new interface, or moves an interface from one location to another, it typically wants to signal the change to its peers. The HLP/HIP protocol includes the Readdressing Packet (REA) for this purpose. Naturally, all the REA packets must be secured with the Host Identity Security Context, and any new addresses must go through the Return Routability test.

## 4.3 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 Analysis

In Sections 3 and 4 we have presented the new architecture and discussed its security properties. In this section we return to a number of points that were briefly mentioned in Section 1 but that can be thoroughly discussed only now. We start in Section 5.1 by rehersing our claim that instead of adding functionality to the top of the internetworking layer it seems to be beneficial to add a completely new layer. Section 5.2 discusses site multi-homing and Section 5.3 how the architecture can be applied to optimize the locality of signalling traffic. Finally, Section 5.4 discusses user and process mobility.

## 5.1 New layer or not

In Section 1.1 we argued that a new name space is clearly needed, but that in addition to that we also need a new protocol layer. The latter need is more questionable, and therefore we want to return to it for a moment.

Opening an HLP/HIP based connection is a fairly heavy operation. It requires two round trips across the network, and involves public key cryptography. Thus, HLP/HIP is clearly unsuitable for light weight transac-

tions where no end-point based security is needed. DNS queries may be considered the prime example of this; besides, since the HLP/HIP identifiers and addresses of the hosts are stored in the DNS, trying to establish a HIP connection with a DNS server would most probably bring forth a chicken-and-egg problem.

Thus, HLP/HIP provides additional utility values at a cost. That is, a HLP/HIP connection facilitates security, mobility, and multi-homing, on the cost of a relatively heavy set-up. There are clearly services where it is beneficial to pay the cost, e.g. streaming, long lived TCP connections, communicating with a virtual home server, etc. On the other hand, there are also services to whom the cost would be detrimental, DNS queries being the prime example. Consequently, providing HLP/HIP as a separate protocol layer that applications may or may not use seems like the right approach.

## 5.2 Site multi-homing issues

In this paper we have almost solely considered end-host multi-homing. As we mentioned in Section 2.2, limited site multi-homing may not be that different from end-host multi-homing. That is, if site multi-homing is implemented by distributing several externally visible IP addresses to the hosts at the site, the multi-homing problems can be essentially solved by considering each of the hosts being separately multi-homed.

Another approach to site-multihoming is to tackle it from the routing infrastructure point of view. As discussed e.g. by Chiappa [18], it is possible to distribute the IP addresses in such a way that the routing table explosion is avoided even under heavy site multi-homing. However, such an approach would essentially require rethinking the whole address assignment approach, and most probably necessitate the development of new protocols to address the practical problems involved.

## 5.3 Amount and locality of signalling traffic

In Sections 3.3 and 4.3 we discussed the packet forwarding function, and noted that one potential benefit from it is the ability to optimize signalling traffic. That is, by introducing a packet forwarding agent at or close the path that packets take between a mobile node and its peer, we may limitate the signalling traffic between the mobile node and the packet forwarding agent only. Thus, the packet forwarding agent essentially hides the mobile node's movements as long as the mobile node stays "behind" it. In Hierarchical MIPv6 [14] terms this function is performed by a Mobile Anchor Point (MAP).

The basic mechanism utilized in our architecture is structurally identical to that of HMIPv6. What makes our architecture different is its generality. The packet forwarding agent providing the mobility anchor point service is no different from packet forwarding agents used for other purposes. Consequently, the same infrastructureless security solutions can be used here than in other cases (see Section 4.3). It suffices that there exists a protocol that allows a mobile node to locate packet forwarding agents that are likely to be useful from the signalling optimization point of view.

## 5.4 User and process mobility

Without going too deep, we also want to note that the separator of locators and end-point identifiers indirectly facilitates user mobility and process migration. Since our architecture allows a node to host several end-point identifiers, the identifiers can be given different roles. For example, some of the identifiers may be associated with a user or a distinct process group. Given sufficient means for transferring cryptographic keys, process states and communication contexts (migration), the architecture makes it possible to painlessly move processes from one device to another. That, in turn, facilitiates user mobility when combined with suitable user interface means.

Thus, user mobility and process migration is easily supported in our architecture from the communications point of view. If a host can save all the communication state associated with a given end-point identifier, and another host can restore the state, it is easy to continue the actual data transfer without any special means. Even the case where both ends are frozen (serialized) for a period of time and later become online simultaneously is supported, given that at least one end has a long lived packet forwarding agent within the network.

# 6 Conclusions

The focus on this paper has been mobility and multi-homing support by modifying the TCP/IP architecture to include a new name space and a new protocol layer. We have provided one possible design, heavily based on the HLP/HIP approach. Furthermore, we have briefly touched the backward compatibility and API issues.

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.

From our point of view, the main benefit of the new name space and associated protocol layer is added security. Since the end-points are *identified* (and not merely named) with public keys, the architecture makes it easy to create a security context between any two end-points. This, in turn, makes it trivial to address the security requirements inherent to end-host mobility and multi-homing. 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. That is, Network Address Translation (NAT) and other middle box communication (MIDCOM) problems seem to be easier to solve with HLP/HIP than without. For example, the packet forwarding agent function introduced in this paper may be considered as one type of Network Address Translation. 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.

Compared to the Mobile IP architecture, our architecture provides more flexibility and resilience. In our architecture, it is possible to relocate any packet forwarder at any convenient time.In Mobile IP, it is not safe to relovate a Home Agent into a topologiclly different location as long as there are active communications that use the Home Address assigned by that Home Agent. Secondly, our architecture easily supports what would be multiple parallel Home Agents in Mobile IP. That is, it is natural to have multiple parallel semi-permanent packet forwarders that are equal and may reside in completely different locations. In Mobile IP, it is possible to have several parallel Home Agents, but each connection is necessarily bound to only one Home Address, thereby making it impossible for the topologically distinct Home Agents to work as back ups for each other.

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. In Mobile IP, Home Agents are used to solve the reachbility problem, temporary Home Agents to optimize the double jump problem, and Mobile Anchor Points to localize signalling.

From the security point of view, no separate mechanism is needed to secure mobility related signalling since the security inherent to the architecture suffices.

Thus, we have shown how the use of public keys as end-point names leads to a natural security solution for end-host mobility and multi-homing. In the companion papers [5][19] we have discussed the security issues in detail 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, Martti Mäntylä, Alexandru Petrescu, Teemu Rinta-Aho, Göran Schultz, Vesa Torvinen, Zoltan Turanyi, and especially Juha Heinänen 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] 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.

[6] I. Castineyra, N. Chiappa, M. Steenstrup, *The Nimrod Routing Architecture,* RFC1992 (Informational), IETF August 1996.

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

[8] 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.

[9] 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.

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

[11] 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.

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

[13] C. Candolin and P. Nikander, "IPv6 Source Addresses Considered Harmful," in Hanne Riis Nielson (ed.), *Proceedings of NordSec 2001, Sixth Nordoc 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.

[14] Hesham Soliman, Claude Castelluccia, Karim El-Malki, Ludovic Bellier, *Hierarchical MIPv6 mobility management (HMIPv6),* work in progress, Internet Draft draft-ietf-mobileip-hmipv6-05.txt, July 2001.

[15] Carl Williams (Editor), Localized Mobility Management Requirements for IPv6, work in progress, Internet Draft draft-ietf-mobileip-lmm-requirements-00.txt, November 2001.

[16] 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`

[17] 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`

[18] J. Noel Chiappa, discussion at the IETF multi6 mailing list, December 2001, `ftp://ops.ietf.org/pub/lists/2001/multi6.0112`

[19] P. Nikander, J. Ylitalo, and J. Wall, "Integrating Security, Mobility, and Multi-homing in a HIP way", to appear in Proceedings of Symposim on Network and Distributed Systems Security (NDSS'03), San Diego, February 2003, Internet Society 2003.