

HIP Applications

InfraHIP Project

Sasu Tarkoma

Wenpeng Zhou

Miika Komu

20.10.2005

Helsinki Institute for Information Technology
Helsinki University of Technology

Abstract

This report presents a summary of applications and application areas that are envisaged to benefit from the features of the Host Identity Protocol (HIP). The main three features provided by the HIP protocol are security, mobility support, and multi-homing support. We present the following main scenarios, in which HIP is potentially useful: VPN for road warriors, traditional applications including FTP, telnet, and web browsing, ad hoc networks, grid computing, and spam prevention.

Contents

1	Introduction	1
2	HIP APIs	3
2.1	Overview	3
2.2	The HIP Namespace	3
2.3	HIP APIs	5
2.3.1	Legacy HIP API	5
2.3.2	Native HIP API	6
2.3.3	The Referral Problem	7
3	HIP Applications	9
3.1	VPN for Road Warriors	9
3.2	Basic Utilities and Applications	10
3.2.1	FTP	10
3.2.2	Telnet	11
3.2.3	Web Browsing and Web Servers	11
3.3	Personal Mobility	11
4	Versatile Networking: Ad Hoc and Grid Networks	13
4.1	Ad Hoc Networks	13
4.1.1	Ad Hoc Access Network	14
4.1.2	Identity to IP Address Translation at Access Points	15
4.2	Grid Computing	16
4.2.1	Grid Overview	16
4.2.2	The Grid Layers	16
4.2.3	Sharing Resources	17
4.2.4	HIP+Grid	18
5	Spam and Network Control	19
6	Summary	23

CONTENTS

Chapter 1

Introduction

The Host Identity Protocol (HIP) adds a new Host Identity (HI) layer between the transport and network layers [13]. The new layer provides host-to-host authentication, identifying the hosts using their public keys or the hash values of public keys (HITs). The new architecture will provide its intrinsic security and support for IP-layer mobility and multi-homing [14].

Features of the HIP protocol, such as *authentication and encryption, mobility and multi-homing*, and *protection from Denial of Service attacks*, are useful for all applications. In addition, the new cryptographic namespace introduced by HIP may be used to create new novel applications and improve existing ones.

We start by presenting the HIP APIs in more detail in Chapter 2. Then we examine how HIP supports traditional applications, such as *Virtual Private Network (VPN)* for road warriors, *FTP*, *telnet*, and *web browsing* in Chapter 3. Chapter 4 discusses the use of HIP in ad hoc networks and Grid environments. Chapter 5 presents a proposal for using HIP in spam prevention. Finally, in Chapter 6 we present a summary of the report.

Chapter 2

HIP APIs

2.1 Overview

In Host Identity Protocol (HIP) [12], the application layer accesses the transport layer via the socket interface. The application layer uses the traditional TCP/IP IPv4 or IPv6 interface, or the new native HIP API interface provided by the socket layer. It is also possible to use HIP through the IPv4 and IPv6 APIs for legacy purposes. The layering model is illustrated in Figure 2.1. For simplicity, the Internet Protocol security (IPsec) layer has been excluded from the figure.

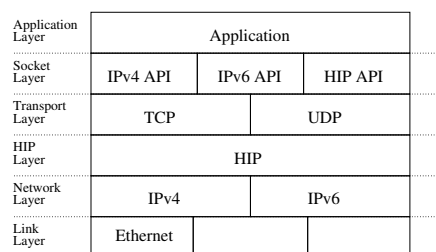


Figure 2.1: The layering model

The HIP layer is as a shim/wedge layer between the transport and network layers. The datagrams delivered between the transport and network layers are intercepted in the HIP layer to see if the datagrams are HIP related and require HIP intervention.

2.2 The HIP Namespace

HIP introduces a new Host Identifier (HI) namespace for the Internet. The HIs are disjoint from the IPv4 and IPv6 namespaces in order to provide

location independent identification of upper-layer endpoints. By decoupling the network-layer identifiers from the upper-layer identifiers, HIP architecture provides a sound foundation on which to build mobility and multi-homing support. The upper layers have stable endpoint identifiers, but the network-layer addresses are allowed to change dynamically.

The endpoints are identified using asymmetric cryptography. A HI is the public key component of an asymmetric key pair. The private key is owned by the endpoint, making impersonating another endpoint very difficult. HIP uses the HIs as Transport Layer Identifiers (TLIs). The locators, i.e. IP addresses, are used only in the network layer. There is a one-to-many binding between a HI and the corresponding locators [17]. As the HI is essentially a variable-sized public key, it is difficult to use in datagram headers. Further, long identifiers are difficult to support in the sockets API because it imposes a limit of 255 bytes to socket address structures. To address these problems, the HIP architecture also includes fixed-size representations of the HI. A Host Identity Tag (HIT) is a 128-bit long hash of the HI. Furthermore, a Local Scope Identifier (LSI) is a 32-bit representation of the HIT.

In the traditional TCP/IP model, connection associations in the application layer are uniquely distinguished by the source IP address, destination IP address, source port, destination port, and transport protocol type. HIP changes this model by using HITs in the place of IP addresses. The HIP model is further expanded in the native HIP API model by using Endpoint Descriptors (EDs) instead of HITs. Now, the application layer uses source ED, destination ED, source port, destination port, and transport protocol type to distinguish between different connection associations.

The ED is used for hiding the representation of endpoints from the applications in the native HIP API. Basically, it is an integer having only local significance, similar to a file or socket descriptor. It acts as a handle or an alias to the corresponding HI on the host.

We believe that using EDs instead of HITs at the application layer has some useful properties. For example, implementing opportunistic base exchange can be easier with EDs. The opportunistic base exchange is initiated to an IP address without the knowledge of the corresponding HIT. In effect, this forces the application to bind to an IP address instead of HITs in the legacy HIP API case, thus complicating the underlying HIP implementation. However, if the application binds to an ED, the networking stack can map it to the corresponding HIT when the HIT is learned during the base exchange. This can be accomplished transparently from the application and without increasing the complexity of the HIP implementation.

The difference between the application and transport layer identifiers is that the transport layer uses HIs instead of EDs. The TLI is named with source HI, destination HI, source port, and destination port at the transport layer. Correspondingly, the HIP layer uses HIs as identifiers. The HIP

security associations are based on source HI and destination HI pairs. The network layer uses IP addresses, i.e. locators, for routing purposes. The network layer interacts with the HIP layer to exchange information about changes in the local interfaces addresses and peer addresses.

A HIP socket is associated with one source and one destination ED, along with their port numbers and protocol type. Multiple EDs and ports can be associated with a single HI. Further, the source HI is associated with a set of network interfaces at the local host. The destination HI, in turn, is associated with a set of destination addresses of the peer.

2.3 HIP APIs

In this section, we describe three APIs for applications to access the HIP namespace. The APIs are described in C-language, although java is supported as well. First, we present the legacy HIP API which is intended as an easy migration path towards HIP enabled applications. Next, we present the native HIP API that allows applications to fully utilize the new namespace and protocol. Finally, we discuss problems related to referrals.

2.3.1 Legacy HIP API

Network applications typically use host names to address peers. Host names have to be resolved to IPv6 addresses from DNS in the resolver library before network connections can be established with peers. In the legacy HIP API [5], the resolver routines have been modified in the implementation so that Domain Name System (DNS) queries prefer to return HITs instead of IPv6 addresses. Otherwise, the legacy API appears like a standard sockets API to the application.

We modified the resolver library to support HIP in two ways. In *transparent mode*, the DNS queries resolve silently to HITs instead of IPv6 addresses. For backwards compatibility, the resolver returns IP addresses if no HITs were found. The greatest benefit of the transparent mode is that it requires no changes in the application. However, a drawback of the transparent mode is that the resolver is not guaranteed to always return HITs. To address this shortcoming, applications can use the resolver in *explicit mode*. Applications pass a flag explicitly to the resolver that enforces the use of HIP by making the resolver return only HITs to the application. Effectively, this means that connections will be established using HIP or not at all. This way, HIP can be used with minimal changes in HIP-aware applications.

An example application using the legacy HIP API is shown in table 2.1. The only modification in the example is the flag to enable the explicit mode.

```
struct addrinfo hints, *res, *try;
char *hello = "hello";
int err, int bytes, sock;

memset(hints, 0, sizeof(hints));
hints.ai_flags = AI_HIP;
hints.ai_family = AF_INET6;
hints.ai_socktype = SOCK_STREAM;

err = getaddrinfo("www.host.org",
                 "echo", &hints,
                 &res);
sock = socket(res->ai_family,
             res->ai_socktype,
             res->protocol);
for (try = res; try; try = try->ai_next)
    err = connect(sock, try->ai_addr,
                 try->ai_addrlen);

bytes = send(sock, hello,
            strlen(hello), 0);
bytes = recv(sock, hello,
            strlen(hello), 0);
err = close(sock);
err = freeaddrinfo(res);
```

Table 2.1: A “hello world” client using the legacy HIP API

2.3.2 Native HIP API

The legacy HIP API requires only minor changes in the applications and therefore it cannot utilize all the features of a HIP-enabled networking stack. Applications requiring more control over the HIP layer can use the native HIP Application Programming Interface (API) [9]. The most significant difference between the legacy and native HIP API is that the native HIP API can use public-key identities in the userspace sockets API. A direct benefit of this is that the users can provide their own public key identifiers to the networking stack. As a result, the identities are not bound to just hosts; they can be bound to users, processes, or groups. Process migration systems [10] may also benefit from this as the HI can be moved along with the process. Additionally, DNS may be used to store public keys instead of HITs [16]. In this case, the explicit public key handling in the native HIP API should become useful.

We propose a *PF_HIP* protocol family to be available in HIP-enabled network stacks. HIP-aware applications use the existing transport layer sockets API and specify this new protocol family when creating sockets. By creating this kind of a HIP-enabled socket, an application can detect whether HIP is supported on the local host. Similarly, an application can detect HIP support in a peer host by resolving the EDs of the peer. If the peer does not support HIP, the resolver returns an empty set.

The syntax of the native HIP API similar to legacy HIP that was illustrated in table 2.1. However, the most crucial differences are the use of *AF_HIP* instead of *AF_INET*, and a new socket structure for EDs. Also, the resolver function is named differently, but used in similar way as the legacy HIP API resolver [9].

An application can control the HIP layer better using the native HIP API than the legacy HIP API. For example, anonymous identities, opportunistic HIP, and certificate handling can be supported using the native HIP API. Quality of service related attributes can also accessed through the native HIP API to allow the simultaneous use of multiple IP flows. This enables applications to benefit from soft-handover strategies or to select a data path depending on the available QoS.

2.3.3 The Referral Problem

HIP introduces a new address space for the transport layer. Basically, the address space is flat although it is possible to use type 2 [12, 16] HITs that contain a domain prefix. Using the prefix, HITs can be resolved to IP addresses from the DNS. However, the problem with this approach is that it has some security implications due to the increased probability of HIT collisions. As a consequence, we may need to have full-length type 1 HITs [12, 16] in the future.

The problem with type 1 HITs is that they are not resolvable to IP addresses in the current DNS infrastructure. This problem occurs, for example, in applications that cache HITs and later try to use them as *callbacks* or pass them to a third party to be used as *referrals* [18]. In such a case, the networking stack of the host cannot route the data traffic requested by the application if the application provides only a HIT.

There are at least to three ways to solve this problem. One way is to modify the DNS infrastructure to support type 2 HIT lookup. The second way is to use an overlay based on a flat namespace such as Internet Indirection Infrastructure (i3) [25] to support resolving of HITs. Third, the overlay can also be used for packet routing, at least for the initial HIP signaling [15].

Chapter 3

HIP Applications

3.1 VPN for Road Warriors

The term *road warrior* is often used to describe a remote worker who wants to access a corporate intranet from various locations and with unknown dynamic IP addresses. Traditionally, the intranet access is arranged using various tunneling solutions and the tunnel needs to be re-established when the mobile host's IP address changes.

HIP, as a protocol that integrates security, mobility, and multi-homing, meets the requirements for the basic road warrior VPN scenario. A road warrior can manually configure the necessary HIs to a server. The server maintains a database of HIs and their associated access rights. During the *base exchange*, the server authenticates the end host HI. Given that the road warrior has sufficient access rights, an end-to-end tunnel is established with the intranet server. The end user does not need to worry about IP address changes due to mobility, multi-homing, or network renumbering.

The challenges for the VPN application scenario include various "road blocks" present in the Internet, such as NATs and firewalls. NAT traversal and firewalls are currently being addressed to support seamless end-to-end communication.

Many of today's access networks are behind a NAT. Moreover, most of the NATs are not very intelligent. Consequently, legitimate incoming traffic is often rejected by NATs. HIP has been explicitly designed to be middlebox friendly. More specifically, the HIP base exchange and mobility management messages have been designed in such a way that a middlebox can learn the association between Host Identity Tags (HITs) and IPsec Security Parameter Indexes (SPIs). In principle, this allows HIP friendly NAT boxes to learn the necessary associations between SPIs and destination addresses at runtime, allowing them to translate ESP packets in addition to TCP and UDP packets.

In order to resolve the routing asymmetry, more complex NAT devices

may allow HIP hosts behind them to register their HITs, thereby supporting inbound connections. In addition, more security conscious middleboxes can verify the puzzle in the I2 message and thus ensure that the return routability mechanism is correctly completed by the base exchange and mobility management protocols. In addition, NATs can offer subnetted hosts some protection from DoS attacks.

Typically corporate firewalls only allow a small set of predefined data classes to pass through. This may result in the dropping of packets containing legitimate traffic. HIP-enabled firewalls may be used to improve connectivity through firewalls, but this requires changes to firewall software.

Corporate firewalls may be configured with a list of HITs and associated public keys, thus limiting access to authenticated hosts. HIP control packets contain the sender's HIT in the IP header instead of the source IP address. The firewalls must keep a table about HITs and the corresponding public keys. With sender's HITs, the firewalls can find the right public key, and a signature. The firewall may verify the signature with the public key, and thereby learn the identity of the end-hosts and selectively open ESP connections based on SPIs. For outbound connections, this provides more security than most firewalls are able to provide today.

3.2 Basic Utilities and Applications

3.2.1 FTP

The File Transfer Protocol (FTP) uses two separate channels (TCP connections) for communication, one for control and one for data. The data channel can be initiated in two ways. In *passive mode*, the server passes its IP address and port number as a callback to the client using the control channel. Then the client initiates a data channel to the server based on the IP address and port number given by the server. In *active mode*, it is the vice versa: the server initiates the data channel to the IP address and port given by the client.

We experimented with the HIP legacy API and FTP. It is not necessary to modify the FTP software in order to use HIP. HIP benefits include end-to-end security and support for user mobility. However, *callbacks* and *referrals* need to be considered.

The FTP way of passing addresses as callbacks can be considered problematic when HIP is used because HITs are used instead of IP addresses. This requires a way to *resolve* a HIT to a routable IP address. The callback and referral problem is solved, for example, by extending DNS to support reverse lookups or using an overlay, such as *Internet Indirection Infrastructure* to support resolving of HITs [15].

3.2.2 Telnet

The basic telnet networking utility does not support data encryption. The immediate benefit of using Telnet over HIP is improved security. We have ported an IPv6-enabled Telnet client and daemon to use the native HIP API. We configured the native HIP API into the code as a compile-time option. The porting process itself was quite straightforward. As the native HIP API resolver name and related data structure are named differently from their IP-based sockets API correspondents, the porting process consisted mainly of search and replace operations in the source code. The API names are different to emphasize the introduction of the new namespace in the resolver but the syntax is almost identical.

3.2.3 Web Browsing and Web Servers

HIP supports secure web browsing in a transparent fashion if the legacy API is used. The client-side does not require recompilation or changes in browser code. HIP support is also easy to incorporate to a web server [11].

3.3 Personal Mobility

Personal mobility and device personalization are becoming an important part of applications and mass-market devices. Personal mobility occurs when the user changes devices. Personalization is needed to change the user experience on a new device to meet the user's expectations. Almost all recent mobile phones support personalization of the device to accommodate the user's preferences, for example in call settings, buddy-lists, and user interface appearance.

A HI may be used to support personal mobility and device personalization. This is accomplished by associating the HI with a user and using a smartcard or a USB stick to store the HI. Personal mobility takes place when the user inserts the identity storage device containing the HI into a terminal device. The HI can then be used to initiate a HIP connection, and to support mobility and multi-homing. The HI may also be used to locate, download, and synchronize data needed for device personalization, such as device, user interface, and preference profiles. Since the HI is a public cryptographic key it allows authentication of the client as well as confidentiality of the personalization data. Privacy may be ensured by using temporary or blinded identifiers.

We have experimented with a scenario in which the HI is stored in a USB memory stick and is thus relocatable between different machines. The insertion of the USB stick is detected automatically. After detection, the HI is loaded to the HIP kernel module, and then used by applications.

CHAPTER 3. HIP APPLICATIONS

We have demonstrated HIP-based connections in personal mobility for file synchronization and for streaming audio playback.

Chapter 4

Versatile Networking: Ad Hoc and Grid Networks

4.1 Ad Hoc Networks

Ad hoc networks consist of mobile nodes that use wireless communication mechanisms. An ad hoc network is typically infrastructureless and nodes are limited in terms of computation power, battery life, and communication range. The routing environment is highly dynamic and requires special network level routing protocols, which are typically divided into two categories *proactive* and *reactive* protocols.

The most likely uses of ad hoc networks include diverse fields such as industrial settings (factories), home/office environments with personal networks, and military and emergency networks. Ad hoc networks may be classified into three types of networks depending on their size and communication technique: low power sensor networks, small and static networks (for example using Bluetooth), and small-to-medium sized mobile networks. These networks may also be combined with each other and fixed network infrastructure to form more complex and versatile networks. Currently, ad hoc networks are primarily used in military and emergency applications, where a reliable fixed infrastructure may not always exist. In the future, it may well be that the number of ad hoc networks connected to the Internet grows [6].

The motivation for IP routing for MANETs is that the IP technology is standards-based, offers flexible and efficient routing, and interoperability with the Internet. The commercial driver for IP-based ad hoc networking is cost effectiveness.

The *MANET* working group at IETF is standardizing IP routing protocol functionality for wireless routing applications within static and dynamic topologies with node mobility. MANETs are deployed at the edge of an IP infrastructure. *Hybrid mesh* infrastructures, which are mixtures of fixed and

mobile routers, are also in the scope of the MANET specifications. The working group addresses both reactive and proactive protocols. Reactive protocols do not maintain active routings tables, whereas proactive protocols are table-driven.

The *Ad hoc On-Demand Distance Vector* (AODV) is a reactive protocol that offers adaptation to dynamic link conditions, low processing and memory overhead, low network utilization, and determines unicast routes to destinations within the ad hoc network [19]. The *Dynamic Source Routing protocol* (DSR) is also a reactive protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes [7]. DSR allows the network to be self-organizing and self-configuring. The protocol consists of the two main mechanisms of "Route Discovery" and "Route Maintenance". These two mechanisms work together to allow nodes to discover and maintain routes to arbitrary destinations in the ad hoc network. The *Optimized Link State Routing Protocol* version 2 (OLSRv2) is an example of a table-driven proactive ad hoc routing protocol. Nodes exchange topology information with each other regularly [3].

Since the ad hoc environment is radically different from the fixed network environment, it is non-trivial to connect ad hoc networks with the Internet. This stems from the fact that the two environments have different routing protocols and strategies. One of the basic notions in the Internet is that each host is identified through its IP address. With more and more networked devices becoming mobile and/or multihomed, this assumption is no longer valid.

HIP may have uses in ad hoc networking, because it separates the locator from the identity. This allows nodes to identify and authenticate each other in dynamic environments. HITs may also be used as routable addresses in ad hoc network routing protocols that are based on flat addresses. The HI may also be used to bridge nodes between different domains, for example between an ad hoc network and a fixed network.

We note that IPsec is only useful for those ad hoc networks that are based on the IP protocol. IPsec is not necessarily useful in ad hoc networks due to the computation complexity of asymmetric cryptography and the limited computation power and battery life of the nodes. Similarly, the HIP puzzle size must be very small if HIP base exchange is to be used between small, wireless devices.

4.1.1 Ad Hoc Access Network

Ad hoc access networks are ad hoc networks that contain one or several nodes that act as gateways to a fixed network, such as the Internet. The Internet uses hierarchical IP addresses as a basis for routing. Many ad hoc networks routing protocols use flat identities. The ad hoc routing protocol may or may not be based on IP. There are many IP based proposals, such as AODV

and DSR. In addition, distributed hash tables based on flat addresses have been integrated with MANET routing protocols, for example the Pastry algorithm with DSR in [22].

HIP changes the transport level connections by associating them with Host Identities rather than IP addresses. This makes it easier to bridge connections between IP networks and ad hoc access network since the Host Identities can be easily used in many different kinds of networks.

In IP networks the creation of a transport session requires the use of the HIP base exchange, which creates an association between IP addresses and Host Identities for that session. This association is required, because routing in IP networks requires the address information of the recipient. In ad hoc networks, hierarchical addressing is not typically used. With HIP both ad hoc networks and the Internet have a common identity space. This leaves us with the problem of doing the necessary translation between identities and IP addresses.

In order to be able to communicate both within an ad hoc network and fixed-network hierarchical IP networks, the ad hoc node needs to have a dual stack that supports both the ad hoc networking protocol and the IP protocol. HIP can be implemented to support both, which makes the mapping between ad hoc nodes and IP network nodes easier.

4.1.2 Identity to IP Address Translation at Access Points

When a node in the ad hoc network, *A*, wishes to communicate with the node *B*, reachable via the fixed network, it first needs to find a route to *B*. To find *B*, *A* needs to detect if *B* is reachable within the ad hoc network, or reachable through an Access Point (AP). If the node is reachable through the AP, it then sends a route reply that contains *B*'s IP address. After the route to *B* is found, *A* initiates the HIP Base Exchange. The access point passing traffic from the ad hoc network performs address translation to the packets, i.e., it puts its own IP address as the sender address and keeps a record of the identities participating in the exchange (as well as the IP address of the correspondent node). One method for doing this is called SPINAT [24].

Fig. 4.1 and Fig. 4.2 show the base exchange and SPI value exchange. The access point recognizes different flows using the SPI parameter. It uses the parameter to do the necessary translations between the two environments.

In order to make it possible to find a host's location based on its Host Identity, an identity lookup service is needed. If a host in an ad hoc access network wishes to be globally reachable, it needs to publish its location information in such a service. The easiest method for this would be to maintain a route to the closest access point and insert its IP address to the lookup service.

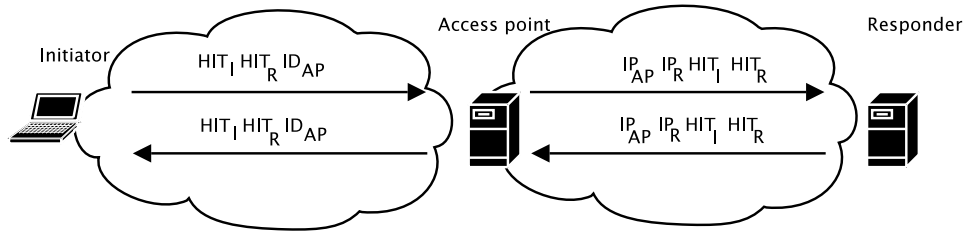


Figure 4.1: Modifications to Base Exchange packets

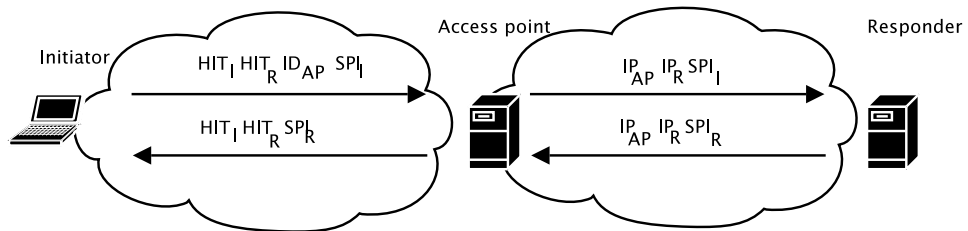


Figure 4.2: Learning SPI at the Access Point

4.2 Grid Computing

4.2.1 Grid Overview

The computation power needed to solve and approximate complex scientific problems is immense and far beyond the capabilities of a single traditional computer. This has led to the development of data grids that link resources across businesses, companies, and academic institutions. This network of computers, the Grid, is then used as a single unified resource to solve problems. The Grid is a particular approach to distributed computing.

4.2.2 The Grid Layers

Figure 4.3 presents the Grid architecture. The *Grid fabric* provides the lowest level of access to actual resources and implements the mechanisms that allow those resources to be utilized. The *Grid connectivity layer* defines communication, security, and authentication protocols required for network transactions between resources. The *Grid resource layer* builds on the connectivity layer to implement protocols that enable the use and sharing of individual resources. There are two fundamental components: information protocols for querying the state of a resource, management protocols to negotiate access to a resource [4].

The highest layer of the structure is the *application layer*, which includes

all different user applications (science, engineering, business, financial), portals and development toolkits supporting the applications.

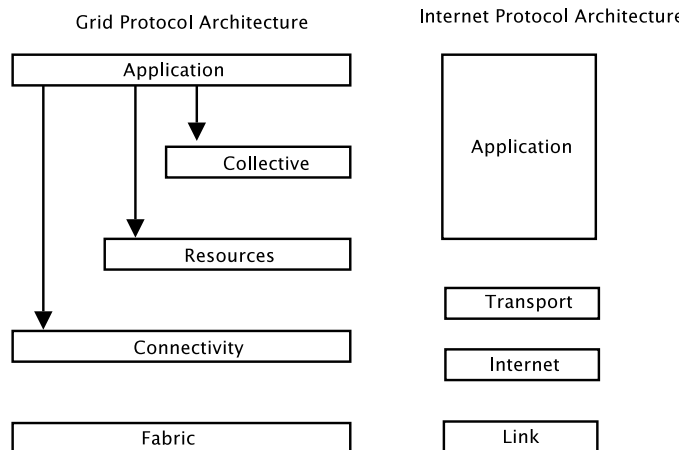


Figure 4.3: The layered Grid architecture

4.2.3 Sharing Resources

The important idea behind the Grid is the sharing of resources. This is more than simple file exchange: it is direct access to remote software, computers and data. It can even give the user access and control of remote sensors, telescopes and other devices. Another important aspect is that the sharing of resources is on a global scale.

Security is a critical aspect of the Grid since there must be a very high level of trust between resource providers and users [23]. A major challenge for the implementation of the Grid comes from this very simple fact: resources are owned by many different people. This means that they exist within different administrative domains, different software platforms and solutions are used, and they are subject to different security and access control policies.

Sharing resources creates some of the most challenging issues for developing Grid applications:

- Access policy - resource providers and users must define clearly and carefully what is shared, who is allowed to share, and the conditions under which sharing occurs.
- Authentication - a mechanism is needed for establishing the identity of a user or resource.
- Authorization - a mechanism is needed for determining whether an operation is consistent with the active sharing policies.

The architecture of the Grid is based on the Internet architecture. Mobility and multi-homing are therefore challenging also for the Grid environment. In addition to host mobility, the Grid environment needs to support also application mobility, in which processes are moved between execution environments [10, 11].

4.2.4 HIP+Grid

Most of the challenges with the Grid and Internet stem from the fact that it is difficult to identify the end user, host, or application. The IP address namespace is not sufficient to solve these challenges. Since HIP is based on a different namespace, the host identifier namespace, it may be used to solve and simplify the challenges.

HIP may be used to provide support for host and user mobility, and tentatively for process mobility [10]. In addition, the HIP identifiers are a good starting point for building access control lists and authorization systems. The mutual authentication performed by the base exchange ensures that entities in the environment can be certain with whom they are exchanging information.

Chapter 5

Spam and Network Control

Today's Internet is the global backbone for digital communication. One of the fundamental applications of the Internet is the exchange of electronic mail. Unfortunately, the phenomenal commercial success of the Internet has boosted the incentive to send unsolicited mail, so called spam, that typically purports the user to buy either real or imaginary products, or simply send out money. Spam has many harmful side-effects, such as filling up server-side mailboxes, taking network capacity and introducing processing overhead at routers and servers, and vexing and frustrating users who cannot find the important mails from the garbage that fills their inboxes.

Spam also introduces serious security and privacy threats. Embedded links and images in email messages can be used to validate and track email addresses. Embedded code in messages can be used to launch various viruses and Trojan horse programs against unprotected users. The emergence of these new ills that plague the Internet has created the necessity to protect end-users from the harmful spam. The current lineup of defense tactics include spam filters, virus, Trojan, and spyware scanners, spammer black lists [8], changes in legislation, quota enforcement using Distributed Hash Tables [1], and sender verification using asymmetric cryptography [20]. Spam filters are classifiers which determine whether an email is spam or not using various filtering techniques, such as Bayesian classifiers.

The crux of the matter is that although spam has prompted a constantly evolving toolkit of remedies, it has not gone away. Spammers use clever techniques to fool spam filters, they change their addresses to keep up with black lists, and changes in legislation are slow to take effect, are difficult and costly to enforce, and change from country to country. The use of asymmetric cryptography can be used to both secure communication and authenticate users, but currently requires that end-users have the motivation and knowledge to use it. Care must be taken so that spam prevention

does not introduce additional limitations and problems for users, such as blacklists. Currently, several spam prevention techniques have been proposed by different stakeholders.

One of the challenges with current Internet email architecture is that it costs very little to send spam. Indeed, many proposals attempt to introduce a cost to spam messages in order to cripple spammers ability to send mass mail. From the network administration viewpoint spam comes in two flavours, *inbound spam* and *outbound spam*. Inbound spam originates from a foreign network and outbound spam is sent to a foreign network. Typically, spam originates from networks infested with zombie machines. A *zombie* machine is a host that has been taken over by spammers or persons working for spammers. Trojans and viruses capture zombie machines and send harmful unsolicited mail.

We propose to tackle the problem of network spam by using the Host Identity Protocol (HIP) [13, 2], being standardized by IETF, to use Host Identities for authenticated SMTP and the cryptographic puzzle in the base exchange (BE) to throttle spammers. The HIP base exchange mechanism has a built-in cryptographic puzzle mechanism to prevent Denial-of-Service (DoS) attacks, which we use to prevent the mass mailing of spam.

The motivation for this proposal is that it is transparent to current routers and SMTP [21] servers. By transparency we mean that HIP does not require large-scale modifications in applications, but a recompilation of code may still be required. The clients and SMTP servers must be HIP enabled, but HIP support can be added without changing routers and without modifying server code other than for addressing related changes. The spam prevention system is applicable in two different environments. First, it may be applied between email clients and SMTP servers within an ISP. Second, it may be applied between SMTP servers within and between ISPs. These two scenarios differ, because the first requires the wide-scale adoption of HIP as an edge technology, and the second requires only that it is used between SMTP servers.

Figure 5.1 illustrates the spam prevention system. The grey boxes in the figure represent components that can be grouped into one physical server, but they are presented separately to highlight modularity. First, the spam source does a DNS lookup for the outbound SMTP server and reads the MX and A records (1). The A record contains the IP address of the rendezvous server. The client starts the HIP base exchange using opportunistic HIP, in which the destination HIT is not required, and sends the first packet of the BE, I1, to RVS (2). If RVS based load balancing is not used, the MX record contains the IP or HIT of the outbound SMTP server or filtering middlebox. The RVS forwards the I1 packet to the outbound filtering middlebox (3). BE is established with this middlebox (4). The outbound middlebox performs filtering and then either drops the message or forwards it to the outbound SMTP server (5). The outbound middlebox and SMTP server are assumed

to be in the same subnet and local network so this communication does not necessarily have to be secure.

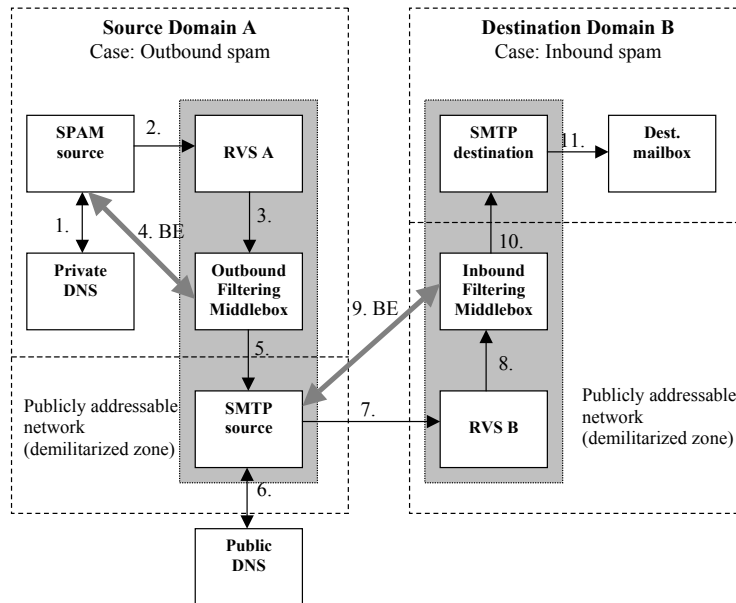


Figure 5.1: HIP spam prevention architecture

After this, the receiving part, the inbound case, of the process begins, and the outbound SMTP server does a DNS lookup to find the destination email server (6), receives the IP address of the RVS, and forwards the I1 packet (7). The destination SMTP server advertises the inbound middlebox in DNS and provides the RVS IP address. The I1 packet is forwarded by RVS (8) and a HIP connection is established with the inbound filter middlebox (9). The inbound filtering middlebox has the same functionality as the outbound middlebox and either accepts the message or drops it (10). The inbound filter can force the sending SMTP server to re-establish the HIP connection and solve a more difficult cryptographic puzzle if spam is encountered. Finally, the message is forwarded to a client mailbox for later retrieval (11).

Chapter 6

Summary

In this report we have presented the following application areas for HIP, namely VPN for road warriors, traditional applications and network utilities, personal mobility, ad hoc networks, grid computing, and spam prevention.

In the VPN for road warriors application area, we discussed how HIP may be used to provide secure and mobility-aware VPN support. Mutual authentication performed by the protocol allows servers to use HIs for client identification. This is useful for access control lists (ACLs) and identity-based personalization. We also examined how HIP-aware NATs and firewalls could enable full universal end-to-end connectivity, which is not possible currently. On the other hand, HIP NAT and firewall traversal requires changes to middleboxes.

In general, HIP applications come in two flavours: legacy and HIP-aware. Applications in the former category do not require modifications, whereas in the latter category applications are modified. FTP and telnet are examples of legacy applications. HIP protocol usage may also be categorized to *client-side* and *server-side*. The mobility support allows seamless services for client systems. Multi-homing support allows highly-available servers and also supports network renumbering.

HIP is also useful in ad hoc networking, because it provides an identity for nodes, and HITs may be used as routable addresses. The HI may be used to bridge ad hoc nodes with the fixed infrastructure nodes without requiring static and globally routable IP addresses for the ad hoc nodes. The limitation of the HIP protocol in ad hoc networking is the possible performance penalty involved in the base exchange phase, because nodes may be resource constrained.

Security and mobility-support are very important for data Grid networks and HIP may also be used in this environment to support mobile nodes and provide mutual authentication between entities. The Grid environment requires also support for process mobility (or migration), which is not directly supported by HIP. The host identities may be used to realize

ACLs also in this case, and the discussion for NATs and firewalls applies as well.

We also proposed a mechanism for using HIP in spam prevention and control. The proposed spam elimination architecture supports both inbound and outbound spam prevention using filtering middleboxes or servers. Connections from clients and SMTP servers to middleboxes are secure and provide a means for introducing a cost for sending spam messages. We envisage that HIP is useful in this context, but further studies are needed to understand the scalability and feasibility of the proposal.

Bibliography

- [1] H. Balakrishnan and D. Karger. Spam-I-am: A Proposal for Spam Control using Distributed Quota Management. In *3rd ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, San Diego, CA, November 2004.
- [2] C. Candolin, M. Komu, M. Kousa, and J. Lundberg. An implementation of HIP for linux. In *Proceedings of the Linux Symposium 2003, Ottawa, Ontario Canada, 23-26 July 2003 pp. 97-105*, July 2003. <http://archive.linuxsymposium.org/ols2003/Proceedings/>.
- [3] T. Clausen. *The Optimized Link-State Routing Protocol version 2*. Internet Engineering Task Force, Aug. 2005. Internet draft, work in progress.
- [4] I. T. Foster. The anatomy of the grid: Enabling scalable virtual organizations. In *Euro-Par '01: Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing*, pages 1–4, London, UK, 2001. Springer-Verlag.
- [5] T.R. Henderson. Using HIP with legacy applications: draft-henderson-hip-applications-00, Feb. 2005. Work in progress. Expires in August 14, 2005.
- [6] IEEE. *Applying Host Identity Protocol to Tactical Networks*, Nov. 2004.
- [7] D. B. Johnson et al. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*. Internet Engineering Task Force, July 2004. Internet draft.
- [8] J. Jung and E. Sit. An Empirical Study of Spam Traffic and the Use of DNS Black Lists. In *Internet Measurement Conference*, Taormina, Italy, October 2004.
- [9] M. Komu. *Native Application Programming Interfaces for the Host Identity Protocol: draft-mkomu-hip-native-api-00*. Internet Engineering Task Force, Sept. 2004. Work in progress. Expires August, 2005.

BIBLIOGRAPHY

- [10] T. Kooponen, A. Gurtov, and P. Nikander. Application mobility with Host Identity Protocol. In *Proc. of NDSS Wireless and Security Workshop*, San Diego, CA, USA, Feb. 2005. Internet Society.
- [11] T. Kooponen, A. Gurtov, and P. Nikander. Virtualizing network connectivity of a migrating virtual machine. 2005. In submission.
- [12] R. Moskowitz and P. Nikander. Host identity protocol architecture: draft-ietf-hip-arch-02.txt, Jan. 2005. Work in progress. Expires in August, 2005.
- [13] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. *Host Identity Protocol*. Internet Engineering Task Force, Oct. 2004. [Internet Draft] <http://www.ietf.org/internet-drafts/draft-ietf-hip-base-01.txt>.
- [14] P. Nikander and J. Arkko. *End-Host Mobility and Multi-Homing with Host Identity Protocol*. Internet Engineering Task Force, Oct. 2004. [Internet Draft] <http://www.ietf.org/internet-drafts/draft-ietf-hip-mm-00.txt>.
- [15] P. Nikander, J. Arkko, and B. Ohlman. Host identity indirection infrastructure (Hi3). In *Proc. of The Second Swedish National Computer Networking Workshop 2004 (SNCNW2004)*, Karlstad, Sweden, Nov. 2004.
- [16] P. Nikander and J. Laganier. Host Identity Protocol (HIP) Domain Name System (DNS) extensions: draft-ietf-hip-dns-01.txt, Feb. 2005. Work in progress. Expires in August, 2005.
- [17] P. Nikander, J. Ylitalo, and J. Wall. Integrating security, mobility, and multi-homing in a HIP way. In *Proc. of Network and Distributed Systems Security Symposium (NDSS'03)*, San Diego, CA, USA, Feb. 2003. Internet Society.
- [18] E. Nordmark. *Multi6 Application Referral Issues*. Internet Engineering Task Force, Jan. 2005. Internet draft, work in progress.
- [19] C. Perkins et al. *Ad hoc On-Demand Distance Vector (AODV) Routing*. Internet Engineering Task Force, July 2003. RFC 3561.
- [20] *PGP FAQ*.
- [21] J. Postel. *Simple Mail Transfer Protocol*, April 2001. <http://www.rfc-editor.org/rfc/rfc2821.txt>.
- [22] H. Pucha, S. M. Das, and Y. C. Hu. Ekta: An efficient dht substrate for distributed applications in mobile ad hoc networks. In *WMCSA*, pages 163–173. IEEE Computer Society, 2004.

- [23] S. Talwar and K. R. Jackson. Overview of security considerations for computational and data grids. In *HPDC '01: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10'01)*, page 439, Washington, DC, USA, 2001. IEEE Computer Society.
- [24] J. Ylitalo and P. Nikander. Blind: A complete identity protection framework for end-points. In *Twelfth International Workshop on Security Protocols, Cambridge, England, Apr. 2004*.
- [25] S. Zhuang, K. Lai, I. Stoica, R. Katz, and S. Shenker. Host mobility using an Internet Indirection Infrastructure. Technical report, University of California at Berkeley, 2002.