HOST IDENTITY PROTOCOL PROXY

Patrik Salmela, Jan Melén
Ericsson Research NomadicLab, Hirsalantie 11, 02420 Jorvas, Finland

Keywords: HIP, identifier-locator split, proxy.

Abstract: The Host Identity Protocol (HIP) is one of the more recent designs that challenge the current Internet architecture. The main features of HIP are security and the identifier-locator split, which solves the problem of overloading the IP address with two separate tasks. This paper studies the possibility of providing HIP services to legacy hosts via a HIP proxy. Making a host HIP enabled requires that the IP-stack of the host is updated to support HIP. From a network administrator's perspective this can be a large obstacle. However, by providing HIP from a centralized point, a HIP proxy, the transition to begin using HIP can be made smoother. This and other arguments for a HIP proxy will be presented in this paper along with an analysis of a prototype HIP proxy and its performance.

1 INTRODUCTION

The current Internet is based on an over 20-year-old architecture. That architecture has flaws - some more serious than others. Many of these issues have been addressed by tools and methods designed to patch the flaws of the architecture. Examples of these new designs are e.g. IPv6 that provides a new larger addresses space in place of the one currently used, and IPsec (Kent (1), 1998) that provides security in the insecure network.

One of the more recent designs is the Host Identity Protocol (Moskowitz (1), 2004). HIP is still being researched and is not yet a complete product and thus not being used in a large scale. Considering that one of the main benefits of using HIP is secured communication, one can assume that HIP might appeal more to companies and organizations rather than the average home computer user. When HIP will begin to be utilized by a larger user group than just the developers and some other interested parties, as it is today, ease of use will be one factor that will affect how well and wide HIP will spread. Enabling HIP in a host requires that the host is updated with a HIP enabled IP-stack. This might be a disadvantage of HIP when a network administrator is considering different methods of protecting the communication to and from the network.

This paper studies the possibility of providing HIP services to hosts without having to modify them. Having legacy hosts communicating with HIP enabled hosts, using HIP, is possible with a HIP proxy. However, providing HIP to hosts via a proxy, with the actual HIP implementation residing outside of the host, puts some restrictions on the network environment. A HIP proxy scenario is shown in Figure 1.

The paper is structured as follows; first some background information will be presented, with the focus on some of the problems of the current architecture. Different solutions for these problems will be presented, including HIP. Next follows a technical view of how HIP works and the reasoning for a HIP proxy. After that, the functionality of a HIP proxy is presented, followed by a look at the design and performance of a prototype HIP proxy. Then we look at how the prototype could be further developed, after which the conclusions are presented.

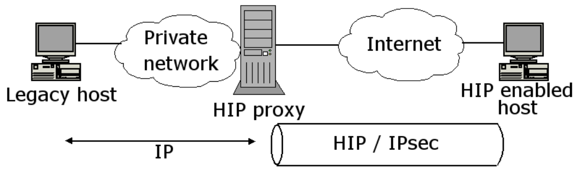

Figure 1: HIP proxy scenario

Salmela P. and Melén J. (2005).
HOST IDENTITY PROTOCOL PROXY.
In *Proceedings of the Second International Conference on e-Business and Telecommunication Networks*, pages 222-230
DOI: 10.5220/0001409002220230
Copyright © SciTePress

2 BACKGROUND

Changing the current Internet architecture is a quite hot topic, and it has been that for some years already. The topic has been discussed in various papers, including the New Arch paper (Braden, 2000) and the Plutarch paper (Crowcroft, 2003). There are many issues with the current architecture that have helped to recognize the need for a change. Maybe the most recognized issues include the lack of support for security by the IP protocol, address space depletion, the heavy load on routers and the overloading of the IP address to serve as both identifier and locator. Additionally, mobile hosts are becoming more common which adds demand for an always better mobility solution.

To some of the aforementioned problems there are already working solutions; users who want security can utilize one of the many available security solutions e.g. IPsec, PGP, SSH or TLS. The utilization of the IPv4 address space has been improved with the help of Classless Inter-Domain Routing (CIDR). Also mobility is possible in the current Internet. Routers are heavily burdened because the size of the IPv4 address does not allow for much address aggregation. IPv6, with its four times bigger address size compared to IPv4, will improve the possibility for address aggregation. However, there is still no widely deployed method that provides an identifier-locator split.

2.1 Why do we need a change

So what is the big deal with using the IP address both as an identifier and a locator? The problem can be spotted by examining how the IP address behaves when a host is changing its topological position in a network, while remembering what qualities are necessary for an identifier and a locator respectively. Consider a host with the IP address IP_A. The locator of the host, i.e. the information used to route packets to the host, is the IP address IP_A. The same information is used to identify the host. If the host moves to another topological position the host has to change its address to the new address IP_A'. When a host now wants to send packets to this host the new IP address, IP_A', is used to route the packets to the host. This means that the locator has changed to match the current location of the host, which is exactly how a locator should function. However, since the IP address serves as both an identifier and a locator the host has now been assigned a new identifier. This change is not welcome since having an identifier that can change frequently makes the identifier useless except for the short timeframe that it stays constant. A true identifier should stay constant, if not forever, at least for a very long time, in the range of years.

Because the notion of an identifier is used in the Internet, it should also fill the requirements set for an identifier. Namely that it is constant and uniquely identifies a host regardless of where in the network the host is located. This makes the IP address an unfit candidate for an identifier. What is needed is another coexistent address space, actually an ``identifier space'', from which hosts are assigned an identity. Another possibility could be something along the lines of what was suggested in the GSE proposal (Crawford, 1999); part of the IP address is used for identifying the host while the rest is used as a locator for the host. In this case the identifier part has to stay constant when the host moves in the network and updates the locator part to match the current location of the host.

2.2 The HIP solution

The Host Identity Protocol is one of the new designs that, amongst other things, target the identifier-locator split. In addition, HIP also provides security, mobility and multi-homing. All the features provided by HIP are based on the solution for the identifier-locator split.

HIP separates the identifier from the locator by introducing a new name space for identifiers. The entities in that set are called Host Identities (HI) and are of variable length. A HI is the public key of an asymmetric key-pair, which is used to provide security in HIP. Because the HIs are of variable length it is difficult to use them as such in HIP, so instead a 128-bit hash over the HI, called a Host Identity Tag (HIT), is used. When operating in an IPv4 network a 32-bit hash over the HI, a Local Scope Identifier (LSI), is used. Because of its length, the LSI cannot be considered to be globally unique. When a HIP enabled host sends a packet to another HIP enabled host the packet is sent to a HIT, or an LSI respectively, but the packet is transported using the locator i.e. the IP address.

The use of HITs and LSIs is made possible by introducing a new layer to the IP-stack. The HIP-layer finds its place between the internetworking layer and the transport layer, and is sometimes referred to as layer 3,5. At the layers above the HIP-layer HITs, or LSIs, are used instead of IP addresses. At the HIP-layer a translation takes place; from HITs or LSIs to IPv6 or IPv4 addresses, or vice versa. In the remaining layers the IP addresses are used. Using HIP, the Host Identifier (HIT or LSI) of a host is always constant as it should be, and the locator can change when the peer moves to another position.

2.3 Other similar solutions

HIP is one of the more complete solutions that provide the identifier-locator split. However, there are also some other proposals that target the same problem. In this subsection three other solutions will be presented: the Forwarding directive, Association, and Rendezvous Architecture (FARA) (Clark, 2003), PeerNet (Eriksson, 2003) and the Internet Indirection Infrastructure (I^3) (Stoica, 2002).

FARA is a framework that can be used when designing a new architecture. The FARA model is divided into two layers; the upper layer contains the communicating entities and the communication endpoints, the lower layer handles the packet forwarding. The communication link between two entities is stateful and is called an Association. Each Association is identified by a locally unique Association ID (AId). When an entity moves its AIds stay constant while the information used to forward the packets to the entity changes. It is easy to draw some parallels between this and how HIP uses constant HIs while the IP address can change to reflect the current position. In the FARA paper (Clark, 2003) HIP is actually suggested as something that could be used in a FARA architecture.

PeerNet is based on peer-to-peer thinking. The hosts are located as leafs in a binary tree, with the path from the root presenting the address of the host. When a new host attaches to the network it asks one of the hosts in its vicinity for an address. The asked host splits its address space into two and assigns one of them to the new node and keeps the other for itself. The hosts also have an identity that stays constant regardless of node movements. PeerNet uses distributed peer-to-peer routing with each host storing some routing information, i.e. identity-to-address mappings. PeerNet is not a ready solution, it does have the identifier-locator split, but security issues have not been addressed.

The I^3 design introduces some new elements to the network, the I^3 servers. To be able to receive packets hosts have to register their identity and current locator into an I^3 server. This is called inserting a trigger. The trigger has a limited lifetime and thus it has to be updated periodically by the host if it wishes to continue to receive packets via it. In I^3 packets are sent to identities and the sent packet searches the I^3 servers for a trigger that matches the destination identity. Once a match is found the destination of the packet is changed for the IP address found in the trigger. By updating the trigger I^3 supports mobility, and by letting multiple hosts register with the same identity a multicast property is achieved. But just as PeerNet, I^3 is not a complete solution. The biggest concern of I^3 is the lack of security. To provide security for I^3 a combination of HIP and I^3, called Hi3, is being researched (Nikander, 2004).

2.4 Problems with having a new architecture

Even if these new designs might sound very good, creating them is only part of the job, getting the design deployed is also a big challenge. Deploying a new architecture is not the same as deploying a new standalone, e.g. security solution. Deploying the design in a small test network which one has full control over is easy, but when the target is a global public network, the Internet, there is not really any good way to get it done. The problem of deploying e.g. HIP is similar to getting IPv6 deployed globally. An ideal solution would be to get all of Internet updated by the flick of a switch, moving from an all IPv4 network to an all IPv6 network in a neglectable time interval. However, this is not possible, not for IPv6 nor HIP. An update of this proportion will proceed incrementally, requiring some sort of compatibility between the new and the old architecture. Deploying HIP is not as difficult as the IPv6 problem since HIP enabled hosts can still communicate with legacy hosts using regular IP. However, to truly benefit from all the features of HIP, it would be desirable that as many hosts as possible were HIP enabled.

3 HIP

To enable HIP in a host the IP-stack of the host has to be updated to a HIP modified one. An asymmetric key-pair has to be generated and the public key will serve as the identity of the host, with hashes of the key resulting in HITs and LSIs. To initiate a HIP connection with another HIP enabled host the HIT of the peer has to be obtained. This can be done from a HIP modified DNS or other similar lookup service.

The creation of a HIP connection between two HIP enabled hosts is called the HIP base exchange (Moskowitz (2), 2004) and it is depicted in Figure 2. When the Initiator wants to establish a connection it sends an I1 packet to the Responder. The packet contains the HIT of the Initiator (HIT_I), and if the HIT of the Responder (HIT_R) has been obtained it is also included in the message. If the Initiator does not know HIT_R it is set to NULL in the I1 packet. This is called opportunistic mode HIP. The I1 packet is actually just an initiation message for the connection.

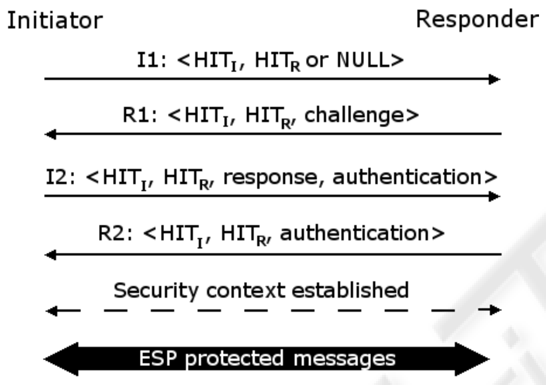
Initiator
Responder
I1: <HIT_I, HIT_R or NULL>
R1: <HIT_I, HIT_R, challenge>
I2: <HIT_I, HIT_R, response, authentication>
R2: <HIT_I, HIT_R, authentication>
Security context established
ESP protected messages

Figure 2: The HIP base exchange

The Responder responds with an R1 packet which contains the HITs used in the I1 packet, the HI of the Responder and a challenge. If the Initiator is attempting opportunistic mode HIP the Responder has now added its HIT to the packet instead of the received NULL HIT. The R1 packet also initiates the Diffie-Hellman (Rescorla, 1999) exchange and gives the preferences of the Responder in respect of which IP Encapsulating Security Payload (ESP) (Kent (2), 1998) mode to use. The supported integrity and encryption algorithms are also presented. The challenge in the packet is a puzzle that the Initiator has to solve to prove that it is serious about creating a connection. The Responder can have in advance prepared R1 packets to ease its load, while the puzzle requires the Initiator to do heavy calculations. This makes connection initiation expensive and is thus a form of Denial of Service (DoS) protection.

When the Initiator has solved the puzzle it sends an I2 packet to the Responder. The packet again contains the two HITs and now also the solution to the puzzle. Also the HI of the Initiator is included, it is encrypted using the selected algorithms and generated keys. Based on the information that the Responder receives in the packet it can decrypt the HI. The Responder also receives the Security Parameter Index (SPI) to use when sending packets to the Initiator.

The last packet of the HIP base exchange, the R2 packet sent to the Initiator, contains the SPI that the Initiator should use along with the two HITs. Similar to all but the first packet of the base exchange, the R2 packet contains a digital signature, and in addition a HMAC (Krawczyk, 1997) calculated over the packet. Besides that, also other consistency checks are done on each packet, including checking that the received HITs are the correct ones. The result of the HIP base exchange is a pair of IPsec ESP security associations (SA). After the base exchange all traffic between the Initiator and the Responder is ESP protected.

The four packets used during the base exchange (I1, R1, I2, R2) are HIP specific packets. Apart from these packets there are also some other HIP specific packets of which the Update packet is the most important one. The Update packet is used for signaling rekeying when the old SA needs to be replaced, e.g. if the ESP sequence number is getting too big. The Update packet is also used for handling location updates by sending location update messages.

The security provided by HIP is basically very similar to IPsec without IKE. The HI of a host, and the corresponding private key, are used for authentication purposes and for negotiating security parameters and SAs. The SAs are established between two HITs, so when sending a packet the SA is located based on the HITs found in the outgoing packet. When receiving a packet the SA is located based on the SPI, and the HITs for the connection are found from the SA.

4 WHY A HIP PROXY

The difficulty of deploying a new architecture was mentioned earlier; all hosts in a global network cannot simultaneously be update to support a new architecture, the migration to a new architecture will take time. HIP does not need to spread to all hosts in the Internet, and it probably never will, but the wider it spreads the more useful HIP is for its users. A HIP proxy that makes it possible for a HIP host to communicate with a legacy host, using HIP between the HIP host and the HIP proxy, could help to promote HIP. The more possibilities there are for using HIP the more appeal it will have. The problem with a HIP proxy is that if it is located in a public network the security features of HIP are rendered useless. The connection between the proxy and the legacy host is not protected in any way. If one would like, some other form of security could of course be applied between the HIP proxy and the legacy host.

To be able to benefit from the security functionality provided by HIP, when using a HIP proxy, the proxy would have to be situated in a secure network. One likely scenario might be a private network, e.g. the internal network of a company. By having a HIP proxy at the border between the private network and the Internet, the users of the private network could contact HIP enabled hosts in the Internet using HIP. Because the private network is considered to be secure the only difference of this scenario, compared to two HIP enabled hosts communicating with each other, is that the legacy host cannot take advantage of all the features provided by HIP, e.g. HIP mobility.

If the hosts of a private network do not need all the features provided by HIP, a HIP proxy might even be considered the preferred alternative compared to enabling HIP in all the hosts. Enabling HIP in all hosts might be considered to generate too much work compared to having a HIP proxy solution. With a static network configuration the work estimates might actually be correct. However, most networks are not static, and having a HIP proxy in a dynamic network will generate excess work in the form of keeping the proxy configurations up-to-date. A HIP proxy is not the preferred solution but it is well suited as a stepping-stone when going from an all legacy network to an all HIP network.

5 THE HIP PROXY PROTOTYPE

As a proof of concept a HIP proxy prototype has been implemented. The implementation was done for FreeBSD 5.2, and tested with the HIP implementation developed at Ericsson Finland (http://hip4inter.net). Besides implementing the HIP proxy application also the kernel of FreeBSD had to be modified; a new feature, divert sockets for IPv6, had to be implemented. To perform its task the proxy utilizes divert sockets and the firewalls (ipfw and ip6fw) of FreeBSD. The network environment where the proxy operates is between two small LANs, one acting as a private network containing the legacy hosts and the other acting as the Internet containing the HIP enabled hosts. If the proxy was to function in one network in which there are both kinds of hosts the legacy hosts would have to be configured to route all their packets via the HIP proxy.

5.1 Functionality of a HIP proxy

When looking at the HIP proxy as a host in the network its task is to serve as the endpoint for HIP associations between itself and HIP enabled hosts. HIP hosts connected via it believe that they are communicating with the legacy host using HIP while the legacy hosts believe that they are communicating with the HIP host using plain IP. For the communicating endpoints the HIP proxy is invisible. The proxy itself can be seen as a host that performs translation between the two communication formats; plain IP and HIP.

When a legacy host wishes to communicate with one of the HIP enabled hosts it queries DNS for the IP address of the peer. The query travels through the HIP proxy and on to a HIP modified DNS in the Internet. The reply contains the IP address and the HIT of the HIP host. When the reply passes the proxy it caches the IP-HIT mapping for future use when it possibly has to initiate a HIP base exchange with the host. The legacy host receives the IP address and can now use it to contact the HIP enabled host. HIP enabled hosts can contact legacy hosts via the proxy if the IP address of the proxy, and the HITs assigned to the legacy hosts, are registered into DNS. Thus a HIP enabled host will receive an IP address and a HIT, as expected, when querying the information about one of the legacy hosts.

When a packet passes through the HIP proxy host the packet must be diverted from its path and sent to the HIP proxy application. If the packet is on its way from a legacy host to a HIP enabled host the proxy checks if there is an SA available for the connection. If a matching SA is found the packet is sent out using the SA. Otherwise the proxy has to initiate the HIP base exchange to establish SAs for the connection. Using the IP-HIT mapping it has

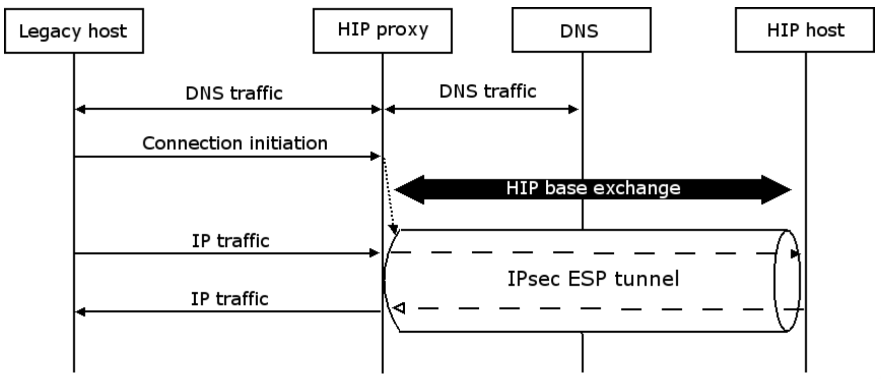

Figure 3: Connection initiation via a HIP proxy

gotten from the DNS query, and the IP address of the legacy host along with the HIT assigned to the legacy host, the proxy can initiate the base exchange. When the HIP association has been established the packet sent by the legacy host can be sent to the HIP enabled host and the communication between the peers can begin. When a packet is received over an SA, from a HIP enabled host, the proxy decrypts the ESP packet and forwards it as a plain IP packet with the IP addresses of the peers. The packet is then sent to the legacy host whose IP address was found based on the destination HIT. The connection initiation is depicted in Figure 3.

When a HIP enabled host initiates a connection to a legacy host it uses the information it has received from DNS. The HIP host believes that it is connecting to the legacy host, although the actual HIP connection is established to the HIP proxy. When the SAs have been established the HIP host begins sending packets over them. The HIP proxy converts the received packets to plain IP packets and forwards them to the correct legacy host.

5.2 The prototype design

The prototype HIP proxy does not function exactly as described in the previous section. We did not have a HIP enabled DNS so the IP-HIT mappings of both the legacy hosts and the HIP enabled hosts were added to a configuration file for the HIP proxy The proxy reads the configuration file and stores the HIT-IP mappings into two linked lists, one for legacy hosts and one for HIP enabled hosts. Apart from the DNS issue the HIP proxy works as described.

To get the received packets diverted to the proxy application we use the IPv4 and IPv6 divert sockets and the firewalls. Basically we tell the firewalls to divert all packets received from the private network except for broadcast packets and other packets that we intuitively know that are not meant for the proxy. This will result in that all connection initiations from the legacy hosts, and the subsequent packets of the connections, go through the proxy. To receive the ESP packets sent from the HIP enabled hosts we tell the proxy to divert all packets that have an address prefix of 01_{bin} for both source and destination addresses. This is a characteristic of HITs; a HIT always has the prefix 01_{bin} (there is also a secondary format for HITs with a 10_{bin} prefix). Even if the packets have IP addresses in the IP header while they travel the Internet, IPsec processing, where the IP addresses are replaced by HITs, happens before the firewall rules are checked. Finally, to allow HIP initiations from the HIP host, we tell the firewall to allow all traffic that uses the HIP protocol, i.e. the packets for the base exchange and the other HIP specific packets such as the Update packet.

The structure of the application is divided into two parts; in the first part the proxy is initialized, the second part consists of a read/write loop where the packets are processed. During the initialization part the configuration file is read and the mappings found in it are recorded. Before a HIP base exchange can begin the Initiator has to have a HIP context for that particular connection. A context consists of the Initiator HIT along with the HIT and IP address of the responder. The prototype creates the needed HIP contexts after the configuration file is read. Before the read/write loop begins the proxy also creates the divert sockets so that it can receive packets.

In the read/write loop the proxy waits for packets diverted to it. Once the proxy receives a packet it examines the source and destination addresses of the packet. Using the two linked lists with HIT-IP mappings the proxy can conclude where the packet is coming from and where it is going to, e.g. from the legacy network to the HIP network. If either of the addresses is not found in the linked lists the proxy cannot process the packet correctly, in that case the packet is forwarded unchanged by the proxy. If mappings for both addresses are found, and both addresses in the packet are found to be either HITs or IP addresses (a mix of one IP and one HIT is not accepted, it indicates an erroneous packet), the proxy changes the IPs for HITs or vice versa. After recalculating the checksums the packet is sent out again. If the packets are going to the legacy host they are forwarded via the output handling to the private network. If the packet is going to one of the HIP hosts it will have HITs as addresses in the IP header. In this case the packet will be sent to IPsec handling. If no SA is found for the specific connection the HIP daemon is signaled to perform the HIP base exchange after which the packet is sent out utilizing the newly created SAs.

Before the read/write loop starts over again the proxy checks if the configuration file should be re-read. This makes it possible to add information about new hosts without restarting the proxy. The prototype uses a very basic method for finding out if the file should be re-read; for each *n* packets the configuration file is re-read. When testing this feature, the value for *n* was set to 10. The value should be adjusted based on how heavy traffic there is through the proxy and the length of the list of hosts entered into the file. With heavy traffic *n* should be increased so that the re-read does not happen very frequently. Also with a long list of hosts *n* should be increased because with a longer list the updating of the linked lists takes longer. A more appropriate solution would be to check if the file has been updated, and only when an update has occurred should the file be re-read.

5.3 Performance

To measure how the HIP proxy prototype performs some tests were conducted. The first test was done to check how having the proxy in the path of the packets affects the round trip times (RTT). First the round trip times for ping6 were measured as an average over 20 packets with the packets going through the proxy but not being processed by it. To get values to compare against the average round trip times were also measured for the case when the packets did not have to go via the proxy, the host with the HIP proxy just forwarded the packets. Finally we measure how the use of the HIP proxy, and having it process packets affected the round trip times. The results from these measurements are presented in Table 1.

It can be concluded from the two first entries that introducing the proxy does add delay; with the proxy we get approximately 12% longer round-trip times. This is something that can be expected since having the packets go via the proxy adds processing on the path. Having to pass a packet to an application in user space, compared to only handling it in kernel space, adds delay.

The last entry in Table 1 concentrates on how applying IPsec ESP to the packets affect the delay. Based on the result we can see that when the HIT-IP mappings are found in the linked lists of the proxy the round-trip time increases approximately by 22% compared to having he proxy sending the packet to output handling without any processing. When we compare the delays of sending packets without using the proxy and the case when the proxy is used and it processes the packets we can see that the increase in delay is approximately 36%. This increase in delay includes both the added delay of having to send the packet to user space, approximately 0,070ms, and the delay that results from performing cryptographic functions, approximately 0,150ms. The by the HIP proxy added delay is mostly a result of doing the cryptographic functions on the data. This is something we cannot affect; if we want security it will cost us time. The total delay added by the proxy is not at an alarming level, and is as such acceptable.

Another interesting aspect of the performance of the proxy is how the amount of entries in the linked lists affects the delay. In the measurements presented in Table 1 there were a total of three entries, two in the HIP hosts list and one in the legacy hosts list. In the next set of measurements we had first 10 then 100 and finally 1000 entries per list. The correct information was situated last in the respective list so that the proxy would have to go through all of the lists. For each packet both the linked lists have to be examined. The results from these measurements are presented in Table 2.

From the measured values we can see that if we add enough entries to the lists it will show in the round-trip times. But since the prototype proxy is not meant for huge networks the delay added by looking up mappings from long lists should not be an issue. The values in Table 2 differ somewhat from the corresponding values in Table 1. The reason for the differing values is that the measurements were performed at different times, so the load on the network was different.

When the proxy reads the host information from a configuration file, as is the case with this prototype, the amount of hosts should be kept small to keep the configuration file manageable. If some automatic updating procedure is implemented it allows for more hosts. Still, the delay caused by having to look up host information from very long lists will sooner or later add too much delay. However, when the amount of hosts configured into the proxy reaches that level it will probably be the amount of traffic that the proxy has to handle that will be the performance bottleneck, not the delay from looking up the correct mappings.

6 FURTHER WORK

In the previous section we concluded that approximately a third of the added delay that results from using a HIP proxy compared to plain IP is a result of the proxy application. This is one aspect of the proxy that could be improved; by moving the application from user space to kernel space the delay induced by the proxy could probably be decreased. Overall the proxy still performs well and as expected. With a small set of hosts the delays are kept at an acceptable level, keeping the RTTs in roughly the same range as for legacy connections. However, one must remember that a HIP proxy is only a solution for a small set of nodes. When the amount of nodes configured into the proxy gets too big, either a second proxy should be introduced, and the load balanced between the proxies, or then the legacy hosts should be made HIP enabled. When a HIP modified DNS is available it will increase the limits of a HIP proxy by being able to dynamically add HIT-IP mappings when they are needed. Also old mappings that are considered obsolete can be deleted since they can be re-fetched from DNS if necessary. The amount of legacy hosts that the proxy can serve will still be a limiting factor.

If the HIP proxy is situated in a public network the security provided by HIP is in effect useless since all the information also travels unencrypted in the network, namely between the proxy and the legacy host. This is quite alright as long as both parties are aware of this. However, when using a HIP proxy the HIP enabled host does not know that it is communicating with a proxy but believes that it is actually communicating with another HIP enabled host. This puts the HIP enabled host at a disadvantage, and it is a problem that needs to be solved; the HIP enabled host must know when it is communicating via a HIP proxy so that it knows that the information it sends might not be secured all the way to the actual endpoint.

A last issue that will be mentioned regarding the HIP proxy is a problem that arises when the HIP host, that is using the services of the HIP proxy, is mobile. When a HIP host is mobile and moves to another location, and thus gets a new locator, it informs its communication parties of its new location. With two HIP enabled hosts this works well. However, when one of the endpoints is a HIP proxy the location update message sent to the proxy modifies established SAs as necessary, but the information does not reach the proxy. If a connection was established between a legacy host and the HIP host before the location change, the connection will continue to work even after the HIP host has moved. If another legacy host now tries to initiate a connection to the mobile host, using its new locator, the connection will not be established since the proxy has not gotten the new locator of the mobile HIP host. This can be solved by updating the proxy configuration file with the new information of the mobile HIP host. This works well if there are no connections established from legacy hosts to the old locator of the HIP host. However, if there still are connections to the old locator the result is that the legacy host using the old locator of the mobile HIP host will begin receiving packets from the HIP host's new locator without knowing that it actually is the same host. A solution for this problem could be that the HIP proxy would keep a record of previous locators of each HIP host, and state information for each connection. Using this information all connections could be maintained. All this of course adds delay to the system. The solution presented

Table 1: How the proxy affects round-trip times

Using proxy	Using HIP	Avg. RTT
No	No	0,624ms
Yes	No	0,698ms
Yes	Yes	0,851ms

Table 2: The effects of serving many hosts

Hosts/list	Avg. RTT
10	0,676ms
100	0,705ms
1000	0,770ms

here is probably not the optimal one and some more research in this area is needed.

7 CONCLUSIONS

The HIP proxy prototype was constructed as a proof-of-concept for a HIP proxy. The proxy performs well and fills its tasks. However, as mentioned in the previous section there are still many areas in which the proxy may, and should, be improved. The preferred solution for using HIP is of course to have HIP enabled hosts. However, a HIP proxy might be a good tool to help HIP get spreading. The HIP proxy prototype described in this paper is probably not something that should be used as such for a HIP proxy. However, it might be a good starting point for developing a new and improved version that better fits the requirements of a HIP proxy.

REFERENCES

Kent (1), Atkins, 1998. Security Architecture for the Internet Protocol. *RFC 2401*

Moskowitz (1), Nikander, 2004. Host Identity Protocol Architecture. *Internet-draft, draft-moskowitz-hip-arch-06 (work in progress).*

Braden, Clark, Shenker, Wroclawski, 2000. Developing a Next-Generation Internet Architecture.

Crowcroft, Hand, Mortier, Roscoe, Warfield, 2003. Plutarch: An Argument for Network Pluralism.

Crawford, Mankin, Narten, Stewart, Zhang, 1999. Separating Identifiers and Locators in Addresses: An Analysis of the GSE Proposal for IPv6. *Internet-draft.*

Clark, Braden, Falk, Pingali, 2003. FARA: Reorganizing the Addressing Architecture.

Eriksson, Faloutsos, Krishnamurthy, 2003. PeerNet: Pushing Peer-to-Peer Down the Stack.

Stoica, Adkins, Ratnasamy, Shenker, Surana, Zhuang, 2002. Internet Indirection Infrastructure.

Nikander, Arkko, Ohlman, 2004. Host Identity Indirection Infrastructure (Hi3).

Moskowitz (2), Nikander, Jokela, Henderson, 2004. Host Identity Protocol. *Internet-draft, draft-ietf-hip-base-00 (work in progress).*

Rescorla, 1999. Diffie-Hellman Key Agreement Method. *RFC 2631.*

Krawczyk, Bellare, Canetti, 1997. HMAC: Keyed-Hashing for Message Authentication. *RFC 2104.*

Kent (2), Atkins, 1998. IP Encapsulating Security Payload (ESP). *RFC 2406.*

HIP for BSD project, http://hip4inter.net *[Referenced 11.02.2005].*