HIP Working Group
Internet-Draft
Expires: August 19, 2008

G. Camarillo
P. Nikander
J. Hautakorpi
Ericsson

February 16, 2008

HIP BONE: Host Identity Protocol (HIP) Based Overlay Networking Environment
draft-camarillo-hip-bone-01.txt

Status of this Memo

Copyright Notice

Abstract

This document specifies a framework to build HIP (Host Identity Protocol)-based overlay networks. This framework uses HIP to perform connection management. Other functions, such as data storage and retrieval or overlay maintenance, are implemented using protocols other than HIP. These protocols are loosely referred to as peer protocols.

Table of Contents

1. Introduction

The Host Identity Protocol (HIP) [I-D.ietf-hip-base] defines a new name space between the network and transport layers. HIP provides upper layers with mobility, multihoming, NAT (Network Address Translation) traversal, and security functionality. HIP implements the so called identifier/location split, which implies that IP addresses are only used as locators, not as host identifiers. This split makes HIP a suitable protocol to build overlay networks that implement identifier-based overlay routing over IP networks, which in turn implement locator-based routing.

The remainder of this document is organized as follows. Section 2 provides background information on HIP. Section 3 describes the HIP BONE (HIP-Based Overlay Networking Environment) framework. Section 4 discusses some of the advantages derived from using the HIP BONE framework. Section 5 briefly explores the relationship of this document to the efforts in the IETF P2PSIP Working Group. Finally, before the customary sections, Section 6 attempts to put the presented proposal into a larger architectural context.

2. Background on HIP

This section provides background on HIP. Given the tutorial nature of this section, readers that are familiar with what HIP provides and how HIP works may want to skip it. All descriptions contain references to the relevant HIP specifications where readers can find detailed explanations on the different topics discussed in this section.

2.1. ID/locator Split

In an IP network, IP addresses typically serve two roles: they are used as host identifiers and locators. IP addresses are locators because a given host's IP address indicates where in the network that host is. IP networks route based on these locators. Additionally, IP addresses are used to identify remote hosts. The simultaneous use of IP addresses as host identifiers and locators makes mobility and multihoming complicated. For example, when a host opens a TCP connection, the host identifies the remote end of the connection by the remote IP address (plus port). If the remote host changes its IP address, the TCP connection will not survive, since the transport layer identifier of the remote end of the connection is gone.

Mobility solutions such as Mobile IP keep the remote IP address from changing so that it can still be used as an identifier. HIP, on the other hand, uses IP addresses as only locators and defines a new

identifier space. This approach is referred to as the ID/locator split and makes the implementation of mobility and multihoming more natural. In the previous example, the TCP connection would be bound to the remote host's identifier, which would not change when the remote hosts moves to a new IP address (i.e., to a new locator). The TCP connection is able to survive locator changes because the remote host's identifier does not change.

2.1.1. Identifier Format

HIP uses 128-bit ORCHIDs (Overlay Routable Cryptographic Hash Identifiers) [RFC4843] as identifiers. ORCHIDs look like IPv6 addresses but cannot collide with regular IPv6 addresses because ORCHID spaces are registered with the IANA. That is, a portion of the IPv6 address space is reserved for ORCHIDs. The ORCHID specification allows the creation of multiple disjoint identifier spaces. Each such space is identified by a separate Context Identifier. The Context Identifier can be either drawn implicitly from the context the ORCHID is used in or carried explicitly in a protocol.

HIP defines a native socket API [I-D.ietf-hip-native-api] that applications can use to establish and manage connections. Additionally, HIP can also be used through the traditional IPv4 and IPv6 TCP/IP socket APIs. Section 2.4 describes how an application using these traditional APIs can make use of HIP. Figure 1 shows all these APIs between the application and the transport layers.

```
              +-----------------------------------------+
              |               Application               |
              +----------------+------------------------+
              | HIP Native API | Traditional Socket API |
              +----------------+------------------------+
              |             Transport Layer             |
              +-----------------------------------------+
```

Figure 1: HIP API

2.1.2. HIP Base Exchange

Before two HIP hosts exchange upper-layer traffic, they perform a four-way handshake that is referred to as the HIP base exchange. Figure 2 illustrates the HIP base exchange. The initiator sends an I1 packet and receives an R1 packet from the responder. After that, the initiator sends an I2 packet and receives an R2 packet from the responder.

```
          Initiator                             Responder
                                   I1
                       -------------------------->
                                   R1
                       <--------------------------
                                   I2
                       -------------------------->
                                   R2
                       <--------------------------
```

Figure 2: HIP base exchange

Of course, the initiator needs the responder's locator (or locators) in order to send its I1 packet. The initiator can obtain locators for the responder in multiple ways. For example, according to the current HIP specifications the initiator can get the locators directly from the DNS [I-D.ietf-hip-dns] or indirectly by sending packets through a HIP rendezvous server [I-D.ietf-hip-rvs]. However, as an architecture HIP is open ended, and allows the locators to be obtained by any means (e.g., from packets traversing an overlay network or as part of candidate address collection in a NAT traversal scenario).

2.1.3. Locator Management

Once a HIP connection between two hosts has been established with a HIP base exchange, the hosts can start exchanging higher-layer traffic. If any of the hosts changes its set of locators, it runs an update exchange [I-D.ietf-hip-mm], which consists of three messages. If a host is multihomed, it simply provides more than one locator in its exchanges. However, if both of the end points move at the same time, or through some other reason both lose track of the peers' currently active locators, they need to resort to using a rendezvous server or getting new peer locators by some other means.

2.2. NAT Traversal

HIP's NAT traversal mechanism is based on ICE (Interactive Connectivity Establishment) [I-D.ietf-mmusic-ice]. Hosts gather address candidates and, as part of the HIP base exchange, hosts perform an ICE offer/answer exchange where they exchange their respective address candidates. Hosts perform end-to-end STUN [I-D.ietf-behave-rfc3489bis] based connectivity checks in order to discover which address candidate pairs yield connectivity.

Even though, architecturally, HIP lies below the transport layer (i.e., HIP packets are carried directly in IP packets), in presence of NATs, HIP sometimes needs to be tunneled in a transport protocol

(i.e., HIP packets are carried by a transport protocol such as UDP).

2.3. Security

Security is an essential part of HIP. The following sections describe the security-related functionality provided by HIP.

2.3.1. DoS Protection

HIP provides protection against DoS (Denial of Service) attacks by having initiators resolve a cryptographic puzzle before the responder stores any state. On receiving an I1 packet, a responder sends a pre-generated R1 packet that contains a cryptographic puzzle and deletes all the state associated with the processing of this I1 packet. The initiator needs to resolve the puzzle in the R1 packet in order to generate an I2 packet. The difficulty of the puzzle can be adjusted so that, if a receiver is under a DoS attack, it can increase the difficulty of its puzzles.

On receiving an I2 packet, a receiver checks that the solution in the packet corresponds to a puzzle generated by the receiver and that the solution is correct. If it is, the receiver processes the I2 packet. Otherwise, it silently discards it.

In an overlay scenario, there are multiple ways how this mechanism can be utilised within the overlay. One possibility to is to cache the pre-generated R1 packets within the overlay and let the overlay directly respond with R1s to I1s. In that way the responder is not bothered at all until the initiator sends an I2 packet, with the puzzle solution. Furthermore, a more sophisticated overlay could verify that an I2 packet as a correctly solved puzzle before forwarding the packet to the responder.

2.3.2. Identifier Assignment and Authentication

As discussed earlier, HIP uses ORCHIDs [RFC4843] as the main representation identifiers. Potentially, HIP can use different types of ORCHIDs as long as the probability of finding collisions (i.e., two nodes with the same ORCHID) is low enough. One way to completely avoid this type of collision is to have a central authority generate and assign ORCHIDs to nodes. To secure the binding between ORCHIDs and any higher-layer identifiers, every time the central authority assigns an ORCHID to a node, it also generates and signs a certificate stating who is the owner of the ORCHID. The owner of the ORCHID then includes the corresponding certificate in its R1 (when acting as responder) and I2 packets (when acting initiator) to prove that it is actually allowed to use the ORCHID and, implicitly, the associated public key.

Having a central authority works well to completely avoid collisions. However, having a central authority is impractical in some scenarios. As defined today, HIP systems generally use a self-certifying ORCHID type called HIT (Host Identity Tag) that does not require a central authority (but still allows one to be used).

A HIT is the hash of a node's public key. A node proves that it has the right to use a HIT by showing its ability to sign data with its associated private key. This scheme is secure due to the so called second-preimage resistance property of hash functions. That is, given a fixed public key K1, finding a different public key K2 such that hash(K1) = hash(K2) is computationally very hard. Optimally, a preimage attack on the 100-bit hash function used in ORCHIDs will take an order of 2^100 operations to be successful, and can be expected to take in the average 2^99 operations. Given that each operation requires the attacker to generate a new key pair, the attack is completely impractical (see [RFC4843]).

HIP nodes using HITs as ORCHIDs do not typically use certificates during their base exchanges. Instead, the use a leap-of-faith mechanism, similar to SSH, whereby a node authenticates somehow remote nodes the first time they connect it and, then, remembers their public keys. While user-assisted leap-of-faith (such as in SSH) can be used to facilitate a human-operated offline path (such as a telephone call), automated leap-of-faith could be combined with a reputation management system to create an incentive to behave. However, such considerations go well beyond the current HIP architecture and even beyond this proposal. For the purposes of the present document, we merely want to point out that architecturally HIP supports both self-generated opportunistic identifiers and administratively assigned ones.

2.3.3. Connection Security

Once two nodes complete a base exchange between them, the traffic they exchange is encrypted and integrity protected. The security mechanism used to protect the traffic is IPsec ESP [I-D.ietf-hip-esp]. However, there is ongoing work to specify how to use different protection mechanisms.

2.4. HIP Deployability and Legacy Applications

As discussed earlier, HIP defines a native socket API [I-D.ietf-hip-native-api] that applications can use to establish and manage connections. New applications can implement this API to get full advantage of HIP. However, in most cases, legacy (i.e., non-HIP aware) applications [I-D.ietf-hip-applications] can use HIP through the traditional IPv4 and IPv6 socket APIs.

The idea is that when a legacy IPv6 application tries and obtains a remote host's IP address (e.g., by querying the DNS) the DNS resolver passes the remote host's ORCHID (which was also stored in the DNS) to the legacy application. At the same time, the DNS resolver stores stores the remote host's IP address internally at the HIP module. Since the ORCHID looks like an IPv6 address, the legacy application treats it as such. It opens a connection (e.g., TCP) using the traditional IPv6 socket API. The HIP module running in the same host as the legacy application intercepts this call somehow (e.g., using an interception library or setting up the host's routing tables so that the HIP module receives the traffic) and runs HIP (on behalf of the legacy application) towards the IP address corresponding to the ORCHID. This mechanism works well in almost all cases. However, applications involving referrals (i.e., passing of IPv6 addresses between applications) present issues, to be discussed in Section 3 below. Additionally, management applications that care about the exact IP address format may not work well with such straigthforward approach.

In order to make HIP work through the traditional IPv4 socket API, the HIP module passes an LSI (Local Scope Identifier), instead of a regular IPv4 address, to the legacy IPv4 application. The LSI looks like an IPv4 address, but is locally bound to an ORCHID. That is, when the legacy application uses the LSI in a socket call, the HIP module intercepts it and replaces the LSI with its corresponding ORCHID. Therefore, LSIs always have local scope. They do not have any meaning outside the host running the application. The ORCHID is used on the wire; not the LSI. In the referral case, if it is not possible to rewrite the application level packets to use ORCHIDs instead of LSIs, it may be hard to make IPv4 referrals work in Internet-wide settings. IPv4 LSIs have been succesfully used in existing HIP deployments within a single corporate network.

3. The HIP BONE Framework

An overlay typically requires three types of operations: overlay maintenance, data storage and retrieval, and connection management. Overlay maintenance operations deal with nodes joining and leaving the overlay and with the maintenance of the overlay's routing tables. Data storage and retrieval operations deal with nodes storing, retrieving, and removing information in or from the overlay. Connection management operations deal with the establishment of connections and the exchange of lightweight messages among the nodes of the overlay, potentially in the presence of NATs.

The HIP BONE framework uses HIP to perform connection management. Data storage and retrieval and overlay maintenance are to be

implemented using protocols other than HIP. For lack of a better name, these protocols are referred to as peer protocols.

3.1. Peer ID Assignment and Bootstrap

Nodes in an overlay are primarily identified by their Peer IDs. (Note that the Peer ID concept here is a peer-layer protocol concept, distinct from the HIP-layer node identifiers. Peer IDs may be long, may have some structure, and may consist of multiple parts.) Overlays typically have an enrollment server that can generate Peer IDs, or at least some part of the Peer ID, and sign certificates. A certificate generated by an enrollment server authorizes a particular user to use a particular Peer ID in a particular overlay. The way users and overlays are identified and the format for Peer IDs are defined by the peer protocol.

The enrollment server of an overlay that were to use plain public keys as Peer IDs could just authorize users to use the public keys and HITs associated to their nodes. This works well as long as the enrollment server is the one generating the public/private key pairs for all those nodes. If the enrollment server authorizes users to use HITs that are generated directly by the nodes themselves, the system is open to a type of chosen-peer-ID attack.

However, in some cases it is impractical to have the enrollment server generate public/private key pairs for devices. In these cases, the enrollment server simply generates Peer IDs whose format is defined by the peer protocol used in the overlay. Since HIP needs ORCHIDs (and not any type of Peer ID) to work, hosts in the overlay will transform their Peer IDs into ORCHIDs, for example, by taking a hash of the Peer IDs or taking a hash of the Peer ID and the public key. That is a similar process to the one a host follows to generate a HIT from a public key. In such scenarios, each host will need a certificate (e.g., in their HIP base exchanges) provided by the enrollment server to prove that they are authorized to use a particular ORCHID in the overlay. Depending on how the certificates are constructed, they typically also need to contain the host's self-generated public key. Depending on how the Peer IDs and public keys are attributed, different scenarios become possible. For example, the Peer IDs may be attributed to users, there may be user public key identifiers, and there may be separate host public key identifiers. Authorisation certificates can be used to bind the different types of identifiers together.

Bootstrap issues such as how to locate an enrollment or a bootstrap server belong to the peer protocol.

3.2. Connection Establishment

Nodes in an overlay need to establish connection with other nodes in different cases. For example, a node typically has connections to the nodes in its forwarding table. Nodes also need to establish connections with other nodes in order to exchange application-layer messages.

As discussed earlier, HIP uses the base exchange to establish connections. A HIP endpoint (the initiator) initiates a HIP base exchange with a remote endpoint by sending an I1 packet. The initiator sends the I1 packet to the remote endpoint's locator. Initiators that do not have any locator for the remote endpoint need to use a rendezvous service. Traditionally, a HIP rendezvous server [I-D.ietf-hip-rvs] has provided such a rendezvous service. In HIP BONE, the overlay itself provides the rendezvous service.

Therefore, in HIP BONE, a node uses an I1 packet (as usual) to establish a connection with another node in the overlay. Nodes in the overlay forward I1 packets in a hop-by-hop fashion according to the overlay's routing table towards its destination. This way, the overlay provides a rendezvous service between the nodes establishing the connection. If the overlay nodes have active connections with other nodes in their forwarding tables and if those connections are protected (typically with IPsec ESP), I1 packets may be sent over protected connections between nodes. Alternatively, if there no such an active connection but the node forwarding the I1 packet has a valid locator for the next hop, the I1 packets may be forwarded directly, in a similar fashion to how I1 packets are today forwarded by a HIP rendezvous server.

Since HIP supports NAT traversal, a HIP base exchange over the overlay will perform an ICE offer/answer exchange between the nodes that are establishing the connection. In order to perform this exchange, the nodes need to first gather candidate addresses. Which nodes can be used to obtain reflexive address candidates and which ones can be used to obtain relayed candidates is defined by the peer protocol.

3.3. Lightweight Message Exchanges

In some cases, nodes need to perform a lightweight query to another node (e.g., a request followed by a single response). In this situation, establishing a connection using the mechanisms in Section 3.2 for a simple query would be an overkill. A better solution is to forward a HIP message through the overlay with the query and another one with the response to the query. The payload of such HIP packet is integrity protected. Nodes in the overlay forward

this HIP packet in a hop-by-hop fashion according to the overlay's routing table towards its destination, typically through the protected connections established between them. Again, the overlay acts as a rendezvous server between the nodes exchanging the messages.

4. Advantages of Using HIP BONE

Using HIP BONE, as opposed to a peer protocol, to perform connection management in an overlay has a set of advantages. HIP BONE can be used by any peer protocol. This keeps each peer protocol from defining primitives needed for connection management (e.g., primitives to establish connections and to tunnel messages through the overlay) and NAT traversal. Having this functionality at a lower layer allows multiple upper-layer protocols to take advantage of it.

Additionally, having a solution that integrates mobility and multihoming is useful in many scenarios. Peer protocols do not typically specify mobility and multihoming solutions. Combining a peer protocol including NAT traversal with a separate mobility mechanism and a separate multihoming mechanism can easily lead to unexpected (and unpleasant) interactions.

5. Relation of HIP BONE with P2PSIP

At IETF 71, the P2PSIP WG will decide whether or not HIP will be included in the solution standardized by the WG. Even if the P2PSIP WG decides not to use HIP, it would be useful that the peer protocol standardized by the P2PSIP WG is kept functionally and specification wise reasonably modular so that the HIP community can use the peer protocol minus the connection management and NAT traversal modules to experiment with HIP-based overlays. Note that, at present, this is the case with several peer protocol proposals.

Another possible outcome would be that the P2PSIP WG makes HIP a mandatory part of P2PSIP. In this case, the HIP WG would need to prioritize all the protocol mechanisms that are not fully specified yet but are needed to implement HIP BONE.

Yet another possible outcome would be that the P2PSIP WG makes HIP an optional part of P2PSIP. There could be several ways to make HIP optional. If HIP is made optional at the overlay level (i.e., all nodes in an overlay either implement HIP or not), there could be P2PSIP-compliant nodes unable to join a particular type of P2PSIP-compliant overlay. If HIP is made optional at the node level (i.e., overlays are made up of nodes some of which implement HIP), there

would be a need for a mechanism to manage hybrid relationships between nodes implementing HIP and nodes not implementing HIP. The complexity of such a mechanism would need to be studied.

6. Architectural Considerations

Architecturally, HIP can be considered to create a new thin "waist" layer on the top of the IPv4 and IPv6 networks; see Figure 3. The HIP layer itself consist of the HIP signalling protocol and one or more data transport protocols; see Figure 4. The HIP signalling packets and the data transport packets can take different routes. In the HIP BONE, the HIP signalling packets are typically first routed through the overlay and then directly (if possible), while the data transport packets are typically routed only directly between the end points.

```
      +--------------------------------------+
      |    Transport (using HITs or LSIs)    |
      +--------------------------------------+
      |                 HIP                  |
      +------------------+-------------------+
      |      IPv4        |       IPv6        |
      +------------------+-------------------+
```

Figure 3: HIP as a thin waist

```
      +------------------+-------------------+
      |  HIP signalling  |  data transports  |
      +------------------+-------------------+
```

Figure 4: HIP layer structure

In HIP BONE, the peer protocol creates a new signalling layer on the top of HIP signalling. It is used to set up forwarding paths for HIP signalling messages. This is a similar relationship that an IP routing protocol, such as OSPF, has to the IP protocol itself. In the HIP BONE case, the peer protocol plays a role similar to OSPF, and HIP plays a role similar to IP. The ORCHIDs are used for forwarding HIP packets according to the information in the routing tables. The peer protocols are used to exchange routing information based on Peer IDs and public keys, and to construct the routing tables.

Architecturally, routing tables are located between the peer protocol

and HIP, as shown in Figure 5. The peer protocol constructs the routing table and keeps it updated. The HIP layer accesses the routing table in order to make routing decisions. The bootstrap of a HIP BONE overlay does not create circular dependencies between the peer protocol (which needs to use HIP to establish connections with other nodes) and HIP (which needs the peer protocol to know how to route messages to other nodes) for the same reasons as the bootstrap of an IP network does not create circular dependencies between OSPF and IP. The first connections established by the peer protocol are with nodes whose locators are known. HIP establishes those connections as any connection between two HIP nodes where no overlays are present. That is, there is no need for the overlay to provide a rendezvous service for those connections.

```
+--------------------------------------+
|             Peer protocol            |
+--------------------------------------+
|             Routing table            |
+--------------------------------------+
|                  HIP                 |
+--------------------------------------+
```

Figure 5: Routing tables

It is possible that different overlays use different routing table formats. For example, the structure of the routing tables of two overlays based on different DHTs (Distributed Hash Tables) may be very different. In order to make routing decisions, the HIP layer needs to convert the routing table generated by the peer protocol into a forwarding table that allows the HIP layer select a next-hop for any packet being routed.

In HIP BONE, the HIP usage of public keys and deriving ORCHIDs through a hash function can be utilised at the peer protocol side to better secure routing table maintenance and to protect against chosen-peer-ID attacks.

The HIP BONE allows quite a lot of flexibility how to arrange the different protocols in detail. Figure 6 shows one potential stack structure.

```
      +-----------------------+--------------+
      | peer protocols        |     media    |
      +------------------+----+--------------+
      | HIP signalling   |   data transport  |
      |                                      |
      +------------------+-------------------+
      | NAT     | non-NAT |                  |
      |                  |                   |
      |      IPv4        |       IPv6        |
      +------------------+-------------------+
```

Figure 6: Example HIP BONE stack structure

7. Security Considerations

TBD.

8. Acknowledgements

HIP BONE is based on ideas coming from conversations and discussions with a number of people in the HIP and P2PSIP communities. In particular, Philip Matthews, Eric Cooper, Alan Johnston, Joakim Koskela, Thomas Henderson, Bruce Lowekamp, and Miika Komu provided useful input on HIP BONE.

9. IANA Considerations

This document does not contain any IANA actions.

10. Normative References

[RFC4843] Nikander, P., Laganier, J., and F. Dupont, "An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers (ORCHID)", RFC 4843, April 2007.

[I-D.ietf-hip-base]
Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, "Host Identity Protocol", draft-ietf-hip-base-10 (work in progress), October 2007.

[I-D.ietf-hip-native-api]
Komu, M., "Native Application Programming Interfaces (APIs) for Host Identity Protocol (HIP)", draft-ietf-hip-native-api-03 (work in progress),

November 2007.

[I-D.ietf-hip-dns]
Nikander, P. and J. Laganier, "Host Identity Protocol (HIP) Domain Name System (DNS) Extensions", draft-ietf-hip-dns-09 (work in progress), April 2007.

[I-D.ietf-hip-rvs]
Laganier, J. and L. Eggert, "Host Identity Protocol (HIP) Rendezvous Extension", draft-ietf-hip-rvs-05 (work in progress), June 2006.

[I-D.ietf-hip-mm]
Henderson, T., "End-Host Mobility and Multihoming with the Host Identity Protocol", draft-ietf-hip-mm-05 (work in progress), March 2007.

[I-D.ietf-mmusic-ice]
Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-ietf-mmusic-ice-19 (work in progress), October 2007.

[I-D.ietf-behave-rfc3489bis]
Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for (NAT) (STUN)", draft-ietf-behave-rfc3489bis-14 (work in progress), February 2008.

[I-D.ietf-hip-esp]
Jokela, P., "Using ESP transport format with HIP", draft-ietf-hip-esp-06 (work in progress), June 2007.

[I-D.ietf-hip-applications]
Henderson, T., Nikander, P., and M. Komu, "Using the Host Identity Protocol with Legacy Applications", draft-ietf-hip-applications-02 (work in progress), November 2007.

Authors' Addresses

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: Gonzalo.Camarillo@ericsson.com

Pekka Nikander
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: Pekka.Nikander@ericsson.com

Jani Hautakorpi
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: Jani.Hautakorpi@ericsson.com

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).