HIP DHT Interface
draft-ahrenholz-hiprg-dht-03

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on April 10, 2009.
Abstract

This document specifies a common interface for using HIP with a Distributed Hash Table service to provide a HIT-to-address lookup service and an unmanaged name-to-HIT lookup service.

Table of Contents

1. Introduction ............................................. 3
2. The OpenDHT interface .................................... 4
3. HIP lookup services ...................................... 7
   3.1. HIP address lookup .................................. 7
   3.2. HIP name to HIT lookup ............................. 10
4. HDRR - the HIP DHT Resource Record ....................... 13
5. When to use the HIP lookup services ....................... 15
6. Issues with DHT support .................................. 17
7. Security Considerations .................................. 18
8. IANA Considerations ..................................... 19
9. Acknowledgments ........................................ 20
10. References .............................................. 21
Appendix A. Change Log ....................................... 22
   A.1. Changes from Version 02 to 03 ...................... 22
   A.2. Changes from Version 01 to 02 ...................... 22
   A.3. Changes from Version 00 to 01 ...................... 22
Author’s Address ........................................... 23
Intellectual Property and Copyright Statements ............... 24
1. Introduction

The Host Identity Protocol [RFC5201] may benefit from a lookup service based on Distributed Hash Tables (DHTs). The Host Identity namespace is flat, consisting of public keys, in contrast to the hierarchical Domain Name System. These keys are hashed and prefixed to form Host Identity Tags (HITs) which appear as large random numbers. The current DNS system does not provide a suitable lookup mechanism for these flat, random values, and has been heavily optimized for address lookup. DHTs manage such data well by applying a hash function that distributes data across a number of servers. DHTs also feature good support for frequently updating stored values.

One freely available implementation of a DHT is the Bamboo DHT, which is Java-based software that has been deployed on PlanetLab servers to form a free service named OpenDHT. OpenDHT is available via the Internet for any program to store and retrieve arbitrary data. OpenDHT uses a well defined XML-RPC interface, featuring put, get, and remove operations. This document discusses a common interface for HIP to be used with OpenDHT, so that various HIP implementations may leverage lookup services in an interoperable fashion.
2. The OpenDHT interface

OpenDHT is a public deployment of Bamboo DHT servers running on about 150 PlanetLab nodes. While the Bamboo project provides the actual software running on the servers, here we will refer only to OpenDHT, which uses a certain defined interface for the XML-RPC calls. One can run their own Bamboo nodes to set up a private ring of servers, but here we are interested in providing a service for use with multiple, different HIP implementations.

OpenDHT was chosen because it is a well-known, publicly available DHT used within the research community. Its interface features a simple, standards-based protocol that can be easily implemented by HIP developers. This document does not aim to dictate that only the services and servers described here should be used, but is rather meant to act as a starting point to gain experience with these services, choosing tools that are readily available.

OpenDHT stores values using (hash) keys. Keys are limited to 20 bytes in length, and values can be up to 1024 bytes. Values are stored for a certain number of seconds, up to a maximum of 604,800 seconds (one week.) See the OpenDHT website: <http://www.opendht.org/>

Three RPC operations are supported: put, get, and rm (remove). Put is called with key and value parameters, causing the value to be stored using the key as its hash index. Get is called with the key parameter, when you have a key and want to retrieve the value. Rm is called with a hash of the value to be removed along with a secret value, a hash of which was included in the put operation.

The definitions below are taken from <http://opendht.org/users-guide.html>.
The put operation takes the following arguments:

<table>
<thead>
<tr>
<th>field</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>string</td>
</tr>
<tr>
<td>client_library</td>
<td>string</td>
</tr>
<tr>
<td>key</td>
<td>byte array, 20 bytes max.</td>
</tr>
<tr>
<td>value</td>
<td>byte array, 1024 bytes max.</td>
</tr>
<tr>
<td>ttl_sec</td>
<td>four-byte integer, max. value 604800</td>
</tr>
<tr>
<td>secret_hash</td>
<td>optional SHA-1 hash of secret value</td>
</tr>
</tbody>
</table>

The server replies with an integer -- 0 for "success", 1 if it is "over capacity", and 2 indicating "try again".

The get operation takes the following arguments:

<table>
<thead>
<tr>
<th>field</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>string</td>
</tr>
<tr>
<td>client_library</td>
<td>string</td>
</tr>
<tr>
<td>key</td>
<td>byte array, 20 bytes max.</td>
</tr>
<tr>
<td>maxvals</td>
<td>four-byte singed integer, max. value 2^31-1</td>
</tr>
<tr>
<td>placemark</td>
<td>byte array, 100 bytes max.</td>
</tr>
</tbody>
</table>

The server replies with an array of values, and a placemark that can be used for fetching additional values.
The `rm` operation takes the following arguments:

<table>
<thead>
<tr>
<th>field</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>string</td>
</tr>
<tr>
<td>client_library</td>
<td>string</td>
</tr>
<tr>
<td>key</td>
<td>byte array, 20 bytes max.</td>
</tr>
<tr>
<td>value_hash</td>
<td>SHA-1 hash of value to remove</td>
</tr>
<tr>
<td>ttl_sec</td>
<td>four-byte integer, max. value 604800</td>
</tr>
<tr>
<td>secret</td>
<td>secret value (SHA-1 of this was used in put)</td>
</tr>
</tbody>
</table>

The server replies with an integer -- 0 for "success", 1 if it is "over capacity", and 2 indicating "try again".

This is the basic XML-RPC interface provided by OpenDHT. Each "field" from the above tables are XML tags that enclose their corresponding values. Below, specific uses for HIP are suggested, along with values that can be used inside the fields shown above.
3. HIP lookup services

Here an address lookup and HIT lookup service are defined for use with HIP. The address lookup uses a peer’s HIT to discover its current addresses. The HIT lookup uses a text name to discover a peer’s HIT.

\[
\begin{align*}
\text{HDDR} &= \text{get}(\text{HIT KEY}) \\
\text{HIT} &= \text{get}(\text{SHA1("name")})
\end{align*}
\]

First is the address lookup, which uses a HIP DHT Resource Record (HDDR) described in Section 4. Before a HIP association can be initiated (a non-opportunistic initiation), a HIP host needs the peer’s HIT and the current address at which the peer is reachable. Often the HIT will be pre-configured or available via DNS lookup using a hostname lookup [RFC5205]. With HIP mobility [RFC5206], IP addresses may be used as locators that are often subject to change. The Host Identity and the HIT remain relatively constant and can be used to securely identify a host, so the HIT serves as a suitable DHT key for storing and retrieving addresses.

The address lookup service includes the peer’s Host Identity and a signature over the locators. This allows the DHT client or server to validate the address information stored in the DHT.

Finally, the HIT lookup service can be used when legacy DNS servers do not support HIP resource records, or when hosts do not have administrative access to their DNS records. This unmanaged naming service may help facilitate the HIP IRTF experiment.

These services reduce the amount of pre-configuration required at each HIP host. The address of each peer no longer needs to be known ahead of time, if peers also participate by publishing their addresses. If peers choose to publish their HITs with a name, peer HITs also no longer need pre-configuration. However, discovering an available DHT server for servicing these lookups will require some additional configuration.

3.1. HIP address lookup

Given a HIT, a lookup returns the HIP DHT Resource Record (HDDR) for the peer. This interface has publish, lookup, and remove operations.

\[
\begin{align*}
\text{HDDR} &= \text{get}(\text{HIT KEY}) \\
\text{put}(\text{HIT KEY}, \text{HDDR}, [\text{secret}]) \\
\text{rm}(\text{HIT KEY}, \text{HDDR}, \text{secret})
\end{align*}
\]

The HDDR is defined in Section 4. It contains one or more locators.
that the peer wants to publish, and the peer’s Host Identity and signature over the contents.

The HIT_KEY is a portion of the HIT used as a DHT key. [RFC4843] defines the HIT as a Prefix concatenated with 100 bits of hash:

\[
\begin{align*}
  \text{Input} & := \text{any bitstring} \\
  \text{Hash Input} & := \text{Context ID} | \text{Input} \\
  \text{Hash} & := \text{Hash function( Hash Input )} \\
  \text{ORCHID} & := \text{Prefix} | \text{Encode}_100( \text{Hash} )
\end{align*}
\]

The HIT_KEY is the \text{Encode}_100( \text{Hash} ) portion of the above definition. Zero padding is appended to this 100-bit value to fill the length required by the DHT, 160 bits total. The HIT’s ORCHID Prefix is dropped because this would cause uneven distribution of the stored values across the DHT servers.

Address publish

+----------------+----------------------------------------+---------+
| field          | value                                  | data    |
|                |                                        | type    |
+----------------+----------------------------------------+---------+
| application    | "hip-addr"                             | string  |
| client_library | (implementation dependent)             | string  |
| key            | 100-bit HIT_KEY                        | base64  |
|                |                                        | encoded |
| value          | HDRR                                   | base64  |
|                |                                        | encoded |
| ttl_sec        | amount of time HDRR should be valid, or the lifetime of the preferred address | numeric |
| secret_hash    | optional SHA-1 hash of secret value    | base64  |
|                |                                        | encoded |
## Address lookup

| field          | value                                                   | data type            |
|----------------+---------------------------------------------------------+----------------------|
| application    | "hip-addr"                                             | string               |
| client_library | (implementation dependent)                              | string               |
| key            | 100-bit HIT_KEY                                         | base64 encoded       |
| maxvals        | (implementation dependent)                              | numeric string       |
| placemark      | (NULL, or used from server reply)                       | base64 encoded       |

## Address remove (optional)

| field          | value                                                   | data type            |
|----------------+---------------------------------------------------------+----------------------|
| application    | "hip-addr"                                             | string               |
| client_library | (implementation dependent)                              | string               |
| key            | 100-bit HIT_KEY                                         | base64 encoded       |
| value_hash     | SHA-1 hash of HDRR (value used during publish) to remove | base64 encoded       |
| ttl_sec        | old address lifetime                                    | numeric string       |
| secret         | secret value (SHA-1 of this was used in put)            | base64 encoded       |

The application and client_library fields are used for logging in OpenDHT. The client_library may vary between different implementations, specifying the name of the XML-RPC library used or the application that directly makes XML-RPC calls.

The key for both address publish and lookup is the 100-bits of the HIT_KEY as defined above, plus 60-bits of zero padding, base64 encoded [RFC2045]. The value used in the publish and lookup response is the base64 encoded HDRR containing one or more LOCATORs.
The ttl_sec field used with address publish includes the time-to-live, the number of seconds for which the entry will be stored by the DHT, which is set to the number of seconds remaining in the address lifetime.

The secret_hash is an optional field used with address publish, used if the value will later be removed with an rm operation. The secret_hash contains the base64 encoded SHA-1 hash of some secret value known only to the publishing host. Clients SHOULD include the secret_hash and remove outdated values to reduce the amount of data the peer needs to handle.

The max_vals and placemark fields used with address lookup are defined by the get XML-RPC interface. The get operation needs to know the maximum number of values to retrieve. The placemark is a value found in the server reply that causes the get to continue to retrieve values starting at where it left off.

3.2. HIP name to HIT lookup

Given the SHA-1 hash of a name, a lookup returns the HIT of the peer. The hash of a name is used because OpenDHT keys are limited to 20 bytes, so this allows for longer names. Publish, lookup, and remove operations are defined.

\[
\text{HIT} = \text{get} (\text{SHA-1} ("name"))
\]

put (SHA-1 ("name", HIT, [secret]))

rm (SHA-1 ("name", HIT, secret))

**HIT publish**

<table>
<thead>
<tr>
<th>field</th>
<th>value</th>
<th>data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>&quot;hip-name-hit&quot;</td>
<td>string</td>
</tr>
<tr>
<td>client_library</td>
<td>(implementation dependent)</td>
<td>string</td>
</tr>
<tr>
<td>key</td>
<td>SHA-1 hash of name</td>
<td>base64 encoded</td>
</tr>
<tr>
<td>value</td>
<td>128-bit HIT</td>
<td>base64 encoded</td>
</tr>
<tr>
<td>ttl_sec</td>
<td>name lifetime</td>
<td>numeric string</td>
</tr>
<tr>
<td>field</td>
<td>value</td>
<td>data type</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>application</td>
<td>&quot;hip-name-hit&quot;</td>
<td>string</td>
</tr>
<tr>
<td>client_library</td>
<td>(implementation dependent)</td>
<td>string</td>
</tr>
<tr>
<td>key</td>
<td>SHA-1 hash of name</td>
<td>base64 encoded</td>
</tr>
<tr>
<td>maxvals</td>
<td>(implementation dependent)</td>
<td>numeric string</td>
</tr>
<tr>
<td>placemark</td>
<td>(NULL, or used from server reply)</td>
<td>base64 encoded</td>
</tr>
</tbody>
</table>

**HIT lookup**

**HIT remove (optional)**

<table>
<thead>
<tr>
<th>field</th>
<th>value</th>
<th>data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>application</td>
<td>&quot;hip-name-hit&quot;</td>
<td>string</td>
</tr>
<tr>
<td>client_library</td>
<td>(implementation dependent)</td>
<td>string</td>
</tr>
<tr>
<td>key</td>
<td>SHA-1 hash of name</td>
<td>base64 encoded</td>
</tr>
<tr>
<td>value_hash</td>
<td>SHA-1 hash of address value to remove</td>
<td>base64 encoded</td>
</tr>
<tr>
<td>ttl_sec</td>
<td>name lifetime</td>
<td>numeric string</td>
</tr>
<tr>
<td>secret</td>
<td>secret value (SHA-1 of this was used in put)</td>
<td>base64 encoded</td>
</tr>
</tbody>
</table>

The key for both HIT publish and lookup is the SHA-1 hash of the name. The name does not necessarily need to be associated with a valid DNS or host name. It does not need to be related to the Domain Identifier found in HI TLV. OpenDHT limits the keys to 20 bytes in length, so the SHA-1 hash is used to allow arbitrary name lengths.
The value used in the publish and lookup response is the base64-encoded 128-bit HIT. This value is the full 128 binary bits of the HIT, which can be identified as a HIT both by its length and by the ORCHID prefix ([RFC4843]) that it starts with.

The ttl_sec field specifies the number of seconds requested by the client that the entry should be stored by the DHT server, which is implementation dependent. Note that this value may not be honored by the server.

The secret_hash is an optional field used with HIT publish if the value will later be removed with an rm operation. The secret_hash contains the base64 encoded SHA-1 hash of some secret value known only to the publishing host. The max_vals and placemark fields used with the HIT lookup are defined by the get XML-RPC interface.
4. HDRR - the HIP DHT Resource Record

The HIP DHT Resource Record uses the same binary format as HIP packets (defined in [RFC5201].) This packet encoding is used as a convenience, even though this data is actually a resource record stored and retrieved by the DHT servers, not a packet sent on the wire by a HIP protocol daemon. Note that this HDRR format is different than the HIP RR used by the Domain Name System as defined in [RFC5205].

HIP header values for the HDRR:

HIP Header:
   Packet Type = 20 (this value is TBD)
   SRC HIT = Sender’s HIT
   DST HIT = NULL

HIP ( LOCATOR, HOST_ID, HIP_SIGNATURE )

The Initiator HIT (Sender’s HIT, SRC HIT) is set to the HIT that the host wishes to make available using the lookup service. This HIT is the same one used to derive the HIT_KEY used as the DHT key. The Responder HIT (Receiver’s HIT, DST HIT) MUST be NULL (all zeroes) since the data is intended for any host.

The LOCATOR parameter contains the addresses that the host wishes to make available using the lookup service. A host may publish its current preferred IPv4 and IPv6 locators, for example.

The HOST_ID parameter contains the Host Identity that corresponds with the Sender’s HIT. (The encoding of this parameter is defined in section 5.2.8 of [RFC5201].)

The HOST_ID parameter and HIP_SIGNATURE parameter MUST be used with the HDRR so that HIP clients receiving the record can validate the sender and the included LOCATOR parameter. The HIT_KEY used for the DHT key will also be verified against the Host Identity.

The client that receives the HDRR from the DHT response MUST perform the signature and HIT_KEY verification. If the signature is invalid for the given Host Identity or the HIT_KEY used to retrieve the record does not match the Host Identity, the DHT record retrieved MUST be ignored. Note that for client-only verification the DHT server does not need to be modified, so this would work on the existing OpenDHT PlanetLab deployment.

The DHT server could also verify the SIGNATURE and HOST_ID. This document primarily describes using the legacy OpenDHT service. Such
Host Identity and signature verification would require modifications to the DHT server software. Users running their own server can run the Bamboo DHT software, on which OpenDHT is based, with some modifications. The signature in the put needs to be verified using the given Host Identity, and the HIT_KEY provided as the key needs to match this Host Identity. If either signature or HIT verification fails, the put is not recorded into the DHT. Full specification of this HIP-aware server behavior is outside the scope of this draft.
5. When to use the HIP lookup services

Below are some suggestions of when a HIP implementation may want to use the address and HIT lookup services.

To learn of a peer’s HIT, a host might first consult DNS using the peer’s hostname if the DNS server supports the HIP Resource Record defined by [RFC5205]. Sometimes hosts do not have administrative authority over their DNS entries and/or the DNS server is not able to support HIP resource records. Hosts may want to associate other non-DNS names with their HITs. For these and other reasons, a host may use the HIT publish service defined in Section 3.2. The peer HIT may be learned by performing a DHT lookup of such a name.

Once a peer HIT is learned or configured, an address lookup could be performed so that the LOCATORs can be cached and immediately available for when an association is requested. Implementations might load a list of peer HITs on startup, resulting in several lookups that can take some time to complete.

However, cached LOCATORs may quickly become obsolete, depending on how often the peer changes its preferred address. Performing an address lookup before sending the I1 may be needed. At this time the latency of a lookup may be intolerable, and a lookup could instead be performed after the I1 retransmission timer fires -- when no R1 reply has been received -- to detect any change in address.

A HIP host should publish its preferred LOCATORs upon startup, so other hosts may determine where it is reachable. The host needs to periodically refresh its HDRR entry because each entry carries a TTL and will eventually expire. Also, when there is a change in preferred address, usually associated with sending UPDATE packets with included locator parameters, the host should update its HDRR with the DHT. The old HDRR should be removed using the rm operation, if a secret value was used in the put.

Addresses from the private address space should not be published to the DHT. If the host is located behind a NAT, for example, the host could publish the address of its RVS to the DHT if that is how it is reachable. In this case however, a peer could instead simply use the RVS field of the NATted host’s HIP DNS record, which would eliminate a separate DHT lookup.

A HIP host should also publish its HIT upon startup or whenever a new HIT is configured, for use with the HIT lookup service, if desired. The host should first check if the name already exists in the DHT by performing a lookup, to avoid interfering with an existing name-to-HIT mapping. The name-to-HIT binding needs to be refreshed.
periodically before the TTL expires.

A host may want to update its published values periodically, per local policy, more frequently than the specified TTL. The TTL value is only a hint for the server, and the server might remove the published data prior to TTL expiration, for example if it is running out of resources.
6. Issues with DHT support

Each put operation appends the new value to any existing values. If a host does not remove its old HDRR before adding another, several entries may be present. Therefore when performing an address lookup, the last HDRR in the DHT response list should be used and considered to contain the current preferred address. Before performing each put a host should remove its old HDRR data using the rm operation.

In the case of the HIT lookup service, there is nothing preventing different hosts from publishing the same name. A lookup performed on this name will return multiple HITs that belong to different devices. This is an unmanaged free-for-all service, so this issue will not be solved here; it is recommended that a host simply pick another name.

Selecting an appropriate DHT server to use is not covered here. If a particular server becomes unavailable, the connect will timeout and some server selection algorithm should be performed, such as trying the next server in a configured list. OpenDHT does provide a DNS-based anycast service, when you perform a lookup of "opendht.nyuld.net", it will return the two nearest OpenDHT servers.

Because the put and get calls rely on outside servers located across the Internet, operations may have a latency involved that should be considered when using these services with HIP.

The maximum size of 1024 bytes for the value field will limit the maximum size of the Host Identity that may be used within the HDRR.
7. Security Considerations

There are two classes of attacks on this information exchange between host and DHT server: attacks on the validity of the information provided by the DHT to the host (such as a spoofed DHT response) and attacks on the DHT records themselves (such as polluted records for a given key). Without an authenticated and trusted service, not much can be done to prevent these attacks.

For the address lookup based on HIT (Section 3.1), the validity of the DHT response can be checked with the HOST_ID and SIGNATURE parameters in the HDRR. A HIP initiating host can also validate the DHT response after the R1 message is received during a HIP exchange. The Host Identity provided in the R1 can be hashed to obtain a HIT that can be checked against the original HIT. However, the legacy OpenDHT service does not currently prevent an attacker from polluting the DHT records for a known HIT, thereby causing a denial-of-service attack, since server validation is not performed.

Relying solely on client validation may be harmful. An attacker can replay the put packets containing the signed HDRR, possibly causing stale or invalid information to exist in the DHT. If an attacker replays the signed put message and changes some aspect each time, and if the server is not performing signature and HIT validation, there could be a multitude of invalid entries stored in the DHT. When a client retrieves these records it would need to perform signature and HIT verification on each one, which could cause unacceptable amounts of delay or computation.

To protect against this type of attack, the DHT server could be running HIP and requiring client authentication with a HIP association before accepting HDRR puts. Also a HIP-aware DHT server could be developed to perform the HIT and signature verification of each put. This is briefly described in Section 4.

For the HIT lookup based on name (Section 3.2), there are no guarantees on the validity of the HIT. Users concerned with the validity of HITs found in the DHT should simply exchange HITs out-of-band with peers. Including a signature will not help here because the HIT that identifies the Host Identity for signing is not known ahead of time.
8. IANA Considerations

This document has no actions for IANA.
9. Acknowledgments

Thanks to Tom Henderson, Samu Varjonen, Andrei Gurtov, Miika Komu, and Kristian Slavov for providing comments. Samu most notably contributed the resolver packet and its suggested parameters, which became the HDRR here.
10. References


Appendix A. Change Log

A.1. Changes from Version 02 to 03

Added text about TTL expiration, appending zero padding, HIT value usage. Removed text on anonymous bit. Use RFC references.

A.2. Changes from Version 01 to 02

sockaddr address format changed to use HIP DHT Resource Record containing the HIP LOCATOR format. The HIT prefix is dropped before using it as a key. Separate "secure" service was dropped, and signatures made mandatory. Legacy versus hip-aware DHT servers are distinguished. Text packet examples added.

A.3. Changes from Version 00 to 01

Removed the HIT lookup service -- using the LSI as a key to return a HIT as the value -- and added a HIT lookup service using names.

Added support for OpenDHT remove. Changed all occurrences of "Open DHT" to "OpenDHT".

Added the Host Identity and a signature as a secure address lookup service, with text about running a modified OpenDHT server that can verify signed put messages based on Host Identity signatures.
Author’s Address

Jeff Ahrenholz
The Boeing Company
P.O. Box 3707
Seattle, WA
USA

Email: jeffrey.m.ahrenholz@boeing.com