HIP Research Group
Internet-Draft
Intended status: Informational
Expires: January 27, 2011

J. Ahrenholz
The Boeing Company

July 26, 2010

Host Identity Protocol Distributed Hash Table Interface
draft-irtf-hiprg-dht-01

Abstract

This document specifies a common interface for using HIP with a Distributed Hash Table service to provide a name-to-Host-Identity-Tag lookup service and a Host-Identity-Tag-to-address lookup service.

Status of this Memo

Copyright Notice

Table of Contents

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. As the current DNS system has been heavily optimized for address lookup, it may be worthwhile to experiment with other services such as those defined here. DHTs manage such data well by applying a hash function that distributes data across a number of servers. DHTs are also designed to support frequently updating stored values. For an alternative method of using HITs to lookup IP addresses using DNS, see [I-D.ponomarev-hip-hit2ip].

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 was available via the Internet for any program to store and retrieve arbitrary data. OpenDHT used a well defined XML-RPC interface, featuring put, get, and remove operations. OpenLookup, while not implemented as a DHT, is another deployment of open source software compatible with this OpenDHT interface. This document discusses a common way for HIP to use this OpenDHT interface, so that various HIP experimenters may employ lookup services in an interoperable fashion.

This document represents the consensus of the HIP RG. This document is not an IETF product and does not represent a standard.

2. The OpenDHT interface

OpenDHT was a public deployment of Bamboo DHT servers that ran on about 150 PlanetLab nodes, retired in July 2009. While the Bamboo project provided 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. Another service compatible with this interface is OpenLookup. One can run their own Bamboo nodes to set up a private ring of servers.

OpenDHT was chosen because it was 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 the OpenDHT users guide at <http://opendht.org/users-guide.html>.

The put operation takes the following arguments:

field	type
application	string
client_library	string
key	byte array, 20 bytes max.
value	byte array, 1024 bytes max.
ttl_sec	four-byte integer, max. value 604800
secret_hash	optional SHA-1 hash of secret value

The server replies with an integer -- 0 for "success", 1 if it is "over capacity", and 2 indicating "try again". The return code 3 indicates "failure" and is used for a modified OpenDHT server that performs signature and HIT verification, see Section 4.

The get operation takes the following arguments:

field	type
application	string
client_library	string
key	byte array, 20 bytes max.
maxvals	four-byte singed integer, max. value 2^31-1
placemark	byte array, 100 bytes max.

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:

field	type
application	string
client_library	string
key	byte array, 20 bytes max.
value_hash	SHA-1 hash of value to remove
ttl_sec	four-byte integer, max. value 604800
secret	secret value (SHA-1 of this was used in put)

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. The key is a byte array used to index the record for storage and retrieval from the DHT. The value is a byte array of the data being stored in the DHT. The Application and client_library fields are meta-data used only for logging purposes. The ttl_sec field specifies the number of seconds that the DHT should store the value. The secret_hash field allows values to be later removed from the DHT. The maxvals and placemark fields are for retrieving a maximum number of values and for iterating get results.

The return code of 0 "success" indicates a successful put or remove operation. The return code of 1 "over capacity" means that a client is using too much storage space on the server. The return value of 2 "try again" indicates that the client should retry the put operation because a temporary problem prevented the server from accepting the put.

In the sections that follow, specific uses for these DHT operations and their XML fields are suggested for use with HIP.

3. HIP lookup services

This draft defines a HIT lookup and address lookup service for use with HIP. The HIT lookup uses a text name to discover a peer's HIT. The address lookup uses a peer's HIT to discover its current addresses.

The two lookups are defined below:

```
HDRR([CERT]) = get(SHA-1("name"))
HDRR(LOCATOR, SEQ, HOST_ID, [CERT], HIP_SIG) = get(HIT_KEY)
```

Both services use a HIP DHT Resource Record (HDRR) described in Section 4. This is a wrapper around data contained in TLVs, similar to a HIP control packet. The data contained in each HDRR differs between the two services.

The HIT lookup service returns the Host Identity Tag of a peer given a name. The name could be the FQDN, hostname, or some other alias. This HIT is found in the Sender's HIT field of the HDRR. The HIT is the hash of the public-key based Host Identity as described in [RFC5201]. There are no security properties of the name, unlike the HIT. An optional certificate may be included in the record, for validating the name, providing some measure of security. Which certificates to consider trusted is a policy issue. This service is intended for use when legacy DNS servers do not support HIP resource records, or when hosts do not have administrative access to publish their own DNS records. Such an unmanaged naming service may help facilitate experimentation.

The address lookup returns a locator and other validation data in the HDRR for a given HIT. Before a HIP association can be initiated (not in opportunistic mode), a HIP host needs to know the peer's HIT and the current address at which the peer is reachable. Often the HIT will be pre-configured, available via DNS lookup using a hostname lookup [RFC5205], or retrieved using the HIT lookup service defined in this document. With HIP mobility [RFC5206], IP addresses may be used as locators and may often 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.

These two separate lookups are defined instead of one because the address record is expected to change more frequently, while the name-

to-HIT binding should remain relatively constant. Also the client and server validation of the two records is different, with the HIT lookup using certificates verifying the name and the address lookup using a signature produced by the bearer of a particular Host Identity/HIT.

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 require pre-configuration. However, discovering an available DHT server for servicing these lookups will require some additional configuration.

3.1. 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.

```
HDRR([CERT]) = get(SHA-1("name"))
put(SHA-1("name"), HDRR([CERT]), [SHA-1(secret)])
rm(SHA-1("name"), SHA-1(HDRR), secret)
```

HIT publish

field	value	data type
application	"hip-name-hit"	string
client_library	(implementation dependent)	string
key	SHA-1 hash of a name	base64 encoded
value	HDRR([CERT]), with the HIT to be published contained in the Sender's HIT field of the HDRR, and an optional certificate for validating the name used as the key	base64 encoded
ttl_sec	lifetime for this record, value from 0-604800 seconds	numeric string

secret_hash	optional SHA-1 hash of secret value	base64 encoded

HIT lookup

field	value	data type
application	"hip-name-hit"	string
client_library	(implementation dependent)	string
key	SHA-1 hash of a name	base64 encoded
maxvals	(implementation dependent)	numeric string
placemark	(NULL, or used from server reply)	base64 encoded

HIT remove (optional)

field	value	data type
application	"hip-name-hit"	string
client_library	(implementation dependent)	string
key	SHA-1 hash of a name	base64 encoded
value_hash	SHA-1 hash of HDRR (value used during publish) to remove	base64 encoded
ttl_sec	lifetime for the remove should be greater than or equal to the amount of time remaining for the record	numeric string
secret	secret value (SHA-1 of this was used in put)	base64 encoded

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 HDRR containing the HIT, and an optional certificate. The HIT is stored in the Sender's HIT field in the HDRR header, and is a 128-bit value than can be identified as a HIT both by its length and by the ORCHID prefix ([RFC4843]) that it starts with.

If a certificate is included in this HIT record, the name used for the DHT key should be listed in the certificate. The CERT parameter is defined in [I-D.ietf-hip-cert]. The Common Name (CN) field from the distinguished name (DN) of the X.509.v3 certificate should be used. The server can hash this name to verify it matches the DHT key.

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 or policy dependent.

The secret_hash is an optional field used with HIT publish if the value will later be removed with an rm operation. It is recommended that clients support these rm operations for the values they publish. The secret_hash contains the base64 encoded SHA-1 hash of some secret value known only to the publishing host. A different secret value should be used for each put because rm requests are visible on the network. The max_vals and placemark fields used with the HIT lookup are defined by the get XML-RPC interface.

3.2. HIP address lookup

Given a HIT, a lookup returns the IP address of the peer. The address is contained in a LOCATOR TLV inside the HDRR, along with other validation data. This interface has publish, lookup, and remove operations. A summary of these three operations is listed below. The abbreviated notation refers to the HIP parameter types; for example HIP_SIG is the HIP signature parameter defined by [RFC5201]. The details of these DHT operations is then described in greater detail.

```
HDRR(LOCATOR, SEQ, HOST_ID, [CERT], HIP_SIG) = get(HIT_KEY)
put(HIT_KEY, HDRR(LOCATOR, SEQ, HOST_ID, [CERT], HIP_SIG),
    [SHA-1(secret)])
rm(HIT_KEY, SHA-1(HDRR), secret)
```

The HDRR is defined in Section 4. It contains one or more locators that the peer wants to publish, a sequence number, the peer's Host Identity, an optional certificate, and signature over the contents.

The HIT_KEY is the last 100 bits of the HIT appended with 60 zero bits. This is the portion of the HIT used as a DHT key. The last 100 bits is used to avoid uneven distribution of the stored values across the DHT servers. The first 28 bits is the HIT's ORCHID Prefix defined by [RFC4843], and this prefix is dropped because it is the same for all HITs, which would cause this uneven distribution. Zero padding is appended to this 100-bit value to fill the length required by the DHT, 160 bits total.

Address publish

field	value	data type
application	"hip-addr"	string
client_library	(implementation dependent)	string
key	HIT_KEY	base64 encoded
value	HDRR(LOCATOR, SEQ, HOST_ID, [CERT], HIP_SIG), with the IP address to be published contained in the LOCATOR TLV in the HDRR, along with other validation data	base64 encoded
ttl_sec	amount of time HDRR should be valid, or the lifetime of the preferred address, a value from 0-604800 seconds	numeric string
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	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	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 HIT_KEY as defined above, 160 bits 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. A different secret value should be used for each put because rm requests are visible on the network.

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.

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]. The reason it is different is that it is a different record from a functional point of view: in DNS, the query key is a FQDN, and the return value is a HIT, while here, the query key is a HIT.

HIP header values for the HDRR:

```
  HIP Header:
    Packet Type = 20 DHT Resource Record (this value is TBD)
    SRC HIT = Sender's HIT
    DST HIT = NULL

  HDRR used with HIT lookup:
  HIP ( [CERT] )

  HDRR used with address lookup:
  HIP ( LOCATOR, SEQ, HOST_ID, [CERT], 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. With the HIT lookup service, this is the main piece of information returned by a get operation. For the address 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 only other TLV used with the HIT lookup service is an optional CERT parameter containing a certificate for validating the name that is used as the DHT key. The CERT parameter is defined in [I-D.ietf-hip-cert]. The DHT server can use the certificate to verify that the client is authorized to use the name used for the DHT key, using the hash of the name found in the certificate. The Common Name (CN) field from the distinguished name (DN) of the X.509.v3 certificate should be used. Which certificates the server considers trusted is a policy issue.

The remaining parameters described here are used with the address lookup service.

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 SEQ parameter contains an unsigned 32-bit sequence number, the Update ID. This is typically initialized to zero and incremented by one for each new HDRR that is published by the host. The host should retain the last Update ID value it used for each HIT across reboots, or perform a self lookup in the DHT, since that number may be retained in the DHT records and will determine the preferred address used by peers.

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

The Sender's HIT in the HDRR should correspond with the key used for the lookup and Host Identity verification. The Receiver's HIT should be NULL (all zeroes) in the HDRR header.

When several HDRR records are returned by the server, the client should pick the most recent record as indicated by the Update ID in the SEQ TLV of the HDRR, and perform verification on that record. The order in which records are returned should not be considered.

The DHT server can also verify the SIGNATURE and HOST_ID, with some modifications to the Bamboo DHT software and a new return code with the OpenDHT interface. The signature in the put needs to be verified using the given Host Identity (public key), and the HIT_KEY provided as the lookup key needs to match this Host Identity according to the ORCHID generation method defined by [RFC4843]. If either signature or HIT verification fails, the put is not recorded into the DHT, and the server returns a failure code. The failure code is an additional return code not defined by OpenDHT, with a value of 3.

5. Use cases

Below are some suggestions of when a HIP implementation may want to use the HIT and address 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 3.1. 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 Rendezvous Server (RVS, from [RFC5204]) 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.

When publishing data to the DHT server, care should be taken to check the response from the server. The server may respond with an "over capacity" code, indicating that its resources are too burdened to honor the given size and TTL. The host should then select another server for publishing, or reduce the TTL and retry the put operation.

6. Issues with DHT support

The DHT put operation does not replace existing values. If a host does not remove its old HDRR before adding another, several entries may be present. A client performing a lookup should determine the most recent address based on the Update ID from the SEQ TLV of the HDRR. The order of values returned in the server's response may not be guaranteed. 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. The server may enforce a policy that requires clients to include a certificate when publishing a HIT, and only store HITs with a name that has been authorized by some trusted certificate. Otherwise this is an unmanaged free-for-all service, and 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 formerly provided a DNS-based anycast service, when you performed a lookup of "opendht.nyuld.net", it returned 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 and certificates 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 the server performing some measure of verification, not much can be done to prevent these attacks.

For the HIT lookup based on name (Section 3.1), 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. A certificate may be included with the HIT which guarantees that the name used for the lookup has been authorized by some 3rd party authority. Which certificates are considered trusted is a local policy issue.

For the address lookup based on HIT (Section 3.2), 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, a legacy OpenDHT service without server modifications does 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 should perform signature and HIT verification of each put operation as described in Section 4. Another option would be the server running HIP itself and requiring client authentication with a HIP association before accepting HDRR puts. Further validation would be only accepting HIT and address records from the association bound to the same HIT.

8. IANA Considerations

This document defines a new HIP Packet Type, the HIP Distributed Hash Table Resource Record (HDRR). This packet type is defined in Section 4 with a value of 20.

9. Acknowledgments

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

10. References

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

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

[RFC5201] Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, "Host Identity Protocol", RFC 5201, April 2008.

[RFC5205] Nikander, P. and J. Laganier, "Host Identity Protocol (HIP) Domain Name System (DNS) Extensions", RFC 5205, April 2008.

[RFC5204] Laganier, J. and L. Eggert, "Host Identity Protocol (HIP) Rendezvous Extension", RFC 5204, April 2008.

[RFC5206] Nikander, P., Henderson, T., Vogt, C., and J. Arkko, "End-Host Mobility and Multihoming with the Host Identity Protocol", RFC 5206, April 2008.

[I-D.ponomarev-hip-hit2ip] Ponomarev, O. and A. Gurtov, "Embedding Host Identity Tags Data in DNS", draft-ponomarev-hip-hit2ip-04 (work in progress), July 2009.

[I-D.ietf-hip-cert] Heer, T. and S. Varjonen, "HIP Certificates", draft-ietf-hip-cert-03 (work in progress), April 2010.

Appendix A. Change Log

A.1. Changes from hiprg 00 to 01

Incorporated comments from Ari Keranen: added references to CERT draft and RFC 5204. Added clarifications from OpenDHT user's guide. Simplified description of HIT_KEY. Dropped RFC 2119 language. Added IANA considerations. Other minor corrections and clarifications.

A.2. Changes from Version ahrenholz 06 to hiprg 00

Document name changed to reflect acceptance as a HIPRG document. Text added to introduction about document acceptance.

A.3. Changes from Version 05 to 06

Use the HDRR format as return values for both services. Added optional certificates for both services. Added text about HIP-aware DHT server that validates HITs/signatures. Added SEQ TLV to HDRR, removed text about ordering. Relaxed statement about DNS and referenced draft-ponomarev-hip-hit2ip. Added text describing why HDRR is different than DNS RR. Added text about handling of source/ destination HITs in HDRR. Renamed Section 5 to "Use cases". Added failure code for put. Removed text about servers not honoring TTL. Added text clarifying what OpenLookup is.

A.4. Changes from Version 04 to 05

Reordered Sections 3.2 and 3.1, since the HIT lookup normally occurs before the address lookup. Added text about why two separate lookups are defined. Added text pertaining to the OpenDHT service retiring.

A.5. Changes from Version 03 to 04

Revised text about server treatment of TTL.

A.6. 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.7. 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.8. 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