A Chord-based DHT for Resource Lookup in P2PSIP
draft-zangrilli-p2psip-dsip-dhtchord-00

Status of this Memo

Copyright Notice

Abstract

This document describes how a structured peer-to-peer algorithm is used for resource lookup by a P2PSIP Peer Protocol. Specifically, this work describes how to integrate a DHT based on Chord with dSIP, a proposed P2PSIP Peer Protocol. This document extends the dSIP draft to provide one possible implementation of a pluggable DHT algorithm.

Table of Contents

1. Introduction

This draft describes an overlay algorithm for use by dSIP [I-D.bryan-p2psip-dsip], a proposed P2PSIP Peer Protocol. This overlay algorithm is derived from the Chord algorithm [Chord], but has been adapted to use SIP messages, as specified by dSIP, to communicate between peers in the overlay. Chord is selected because of its simplicity, convergence properties, and general familiarity within the P2P community. This work reflects experience gained in actually building a full commercially available P2PSIP product based on using a Chord-like DHT for resource location in the dSIP protocol, as well as from work/insight gleaned from the P2PSIP mailing list.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Terminology defined in RFC 3261 [RFC3261] is used without definition.

We use the terminology and definitions from the dSIP: A P2P Approach to SIP Registration and Resource Location [I-D.bryan-p2psip-dsip] and the Concepts and Terminology for Peer to Peer SIP [I-D.willis-p2psip-concepts] drafts extensively in this document. Other terms relating to P2P or new to this document are defined when used and are also defined in Definitions (Section 2.1). We suggest reviewing these drafts and the Definitions (Section 2.1) section before reading the remainder of this document..

In many places in this document, 10 hexadecimal digit values are used in examples as SHA-1 hashes. In reality, these hashes are 40 digit. They are shortened in this document for clarity only.

2.1. Definitions

Please also see the dSIP: A P2P Approach to SIP Registration and Resource Location [I-D.bryan-p2psip-dsip] draft and the P2PSIP concepts and terminology [I-D.willis-p2psip-concepts] draft for additional terminology. We do not redefine terms from that draft here.

Chord: A particular algorithm/approach to implementing a DHT, described by Stoica et al. [Chord]. Uses a circular arrangement for the namespace.

Finger Table: The list of peers that a peer uses to send messages to. The finger table contains many entries about peers with similar IDs, and fewer entries about more remote IDs.

Predecessor Peer: Refers to a peer directly before a particular peer in the address space. This does not mean that the predecessor's Peer-ID is one less than the peer, it simply means that there are no other peers in the namespace between the peer and the predecessor peer.

Successor Peer: Refers to a peer directly after a particular peer in the address space. This does not mean that the successor peer's Peer-ID is one more that that peer, rather there are no other peers in the namespace between that peer and the successor peer. Note that the first peer in a finger table is typically also the first successor peer.

3. Background

3.1. Chord

The Chord [Chord] system is one particular popular DHT algorithm. Chord uses a ring-type structure for the peers in the overlay. In this structure, a peer with a hash of 0 would be located adjacent to a peer that hashes to the highest possible hash value. In Chord, resource with Resource-ID k will be stored by the first peer with Peer-ID equal to or greater (mod the size of the namespace) than k, ensuring that every Resource-ID is associated with some peer.

4. Routing Table and Connection Table

Each peer keeps information about how to contact some number of other peers in the overlay. In terms of the overlay network, these are the neighbors of the peer, since they are reachable in one hop. In Chord the peer keeps track of one or more of its immediate predecessor peers, as well as one or more successor peers. The peer also keeps a table of information about other neighbors called a finger table, consisting of peers distributed around the overlay.

Note that dSIP defines a routing table as the set of peers that a peer knows about and uses to send messages to when routing. The routing table is the combination of the predecessor, successor and finger table.

4.1. Finger Table

If the hash has 2^n bits in the range, each peer will keep a "finger table" of pointers to at most n other peers. The ith entry in the

finger table contains a pointer to a peer at least 2^(i) units away in the hash space. The highest finger table entry thus point to a range 1/2 of the way across the hash space, the next highest 1/4, the next 1/8, and the smallest entry points to a range only 1 away in the hash space. The set of peers pointed to by these finger table entries are referred to as the neighbors of the peer, since they can be reached directly.

4.2. Message Routing

Messages are routed by taking advantage of a key property of these finger tables. A peer has more detailed, fine grained information about peers near it than further away, but it knows at least a few more distant peers. When locating a a particular ID (either Resource-ID or Peer-ID), the peer will send the request to the finger table entry with the Peer-ID closest to the desired ID. Because the peer receiving the request has many neighbors with similar Peer-IDs, it will presumably know of a peer with a Peer-ID closer to the ID, and suggests this peer to in response. The request is then resent to this closer peer. The process is repeated until the peer responsible for the ID is located, which can then determine if it is storing the information.

5. Message Syntax

5.1. The DHT-PeerID Header

The routing algorithms used to implement the overlay is specified in the dht-param parameter in the DHT-PeerID header. The format of the DHT-PeerID header is defined in the dSIP [I-D.bryan-p2psip-dsip] draft.

5.1.1. Hash Algorithms

Implementations MUST support the SHA-1 [RFC3174] algorithm, which produces a 160 bit hash value. An implementation MAY rely on a secret initialization vector, key, or other shared secret to use the identifier as an HMAC, from from RFC 2104 [RFC2104] such that no peer may join the overlay without knowledge of the shared secret, however this technique by itself does not protect the overlay against replay attacks. Security Extensions to dSIP [I-D.lowekamp-p2psip-dsip-security] provides information on how to protect against replay attacks and hash algorithms defined in that draft MAY be used in Chord implementations.

5.1.2. DHT Name Parameter

For this protocol, the dht-param token MUST be set to "Chord1.0".

A peer receiving a message with a dht-param other than "Chord1.0" SHOULD reject the message and return a 488 Not Acceptable Here response message.

Examples:
A peer with an SHA-1 hashed Peer-ID of a04d371e24 on IP 192.0.2.1. We include the required algorithm, and overlay as well as the optional expires header parameter.

```
DHT-PeerID: <sip:peer@192.0.2.1;peer-ID=a04d371e24>;algorithm=sha1;
dht=Chord1.0;overlay=chat;expires=600
```

5.2. The DHT-Link Header

The DHT-Link header is used to transfer information about where in the DHT other peers are located. In particular, it is used by peers to pass information about the predecessor, successors, and finger table information stored by a peer.

The linktype and depth values are dependent on the DHT routing algorithm employed by the peer. We define a linktype-token and depth-token in the DHT-Link Header to be used by peers implementing the Chord1.0 DHT algorithm.

```
link-value      = linktype-token depth-token
linktype-token  = "P" / "F" / "S" / other-token
depth-token     = 1*DIGIT
```

and an example, the header might look like (using a shortened digit Peer-ID for clarity):

```
DHT-Link: <sip:peer@192.0.2.1;peer-ID=671a65bf22>;link=S1;expires=600
```

5.2.1. The linktype and depth values

The linktype MUST be one of three single characters, P, S, or F. P MUST be used to indicate that the information provided describes a predecessor of the sending peer. S MUST indicate that the information describes a successor peer, and F MUST indicate that it is a finger table peer from the sending peer.

The depth MUST be a non-negative integer representing which predecessor, successor, or finger table entry is being described.

For predecessors and successors, this MUST indicate numeric depth. In other words, "P1" indicates the peers immediate predecessor, while "S5" would indicate the fifth successor. "P0" or "S0" would indicate the sending peer itself. In the case of finger table entries, the depth MUST indicate the exponent of the offset. Since finger tables point to ranges in the hash table that are offset from the current peer in the hash space by a power of two. That is, finger table entry i points to a range that begins with a peer 2^i away in the hash space, and there are a maximum of k finger table entries, where k is the size of the hash result in bits. For an finger table entry, the depth corresponds to this exponent i. In other words, "F0" would correspond to a finger table entry pointing to the peer for a range starting a distance 2^0 = 1 from the Peer-ID in the hash space, while "F6" would point to peer used to search for resources in a range starting 2^6 = 64 away from the Peer-ID in the hash space.

6. Chord Overlay Algorithm

6.1. Finger Table, Successors, and Predecessors

Each peer MUST have keep track of at least one predecessor peer in its routing table. This predecessor peer CANNOT be set to itself, but CAN be NULL if the current peer is the only peer in the overlay. Each peer MUST also have at least one successor peer in its routing table. This successor peer can be set to itself. Peers MAY keep additional successor and predecessor information in their routing tables for reliability.

Chord recommends keeping a number of finger table entries equal to the size in bits of the hash space, for example 160 for SHA-1. These entries point to the first peer at least 2^i away from the peer, for 0 <= i <= 159, mod 2^160. Essentially, the peer divides the overlay hash circle up into segments, the first being the segment from [2^0-2^1) away from the peer, the second being from [2^1-2^2), the third being from [2^2-2^3), etc., all the way to the segment from [2^158-2^159) away from peer. It then stores an entry in the finger table for the first peer with a Peer-ID greater than or equal to the start of this interval. In this way, the peer has many entries pointing to nearby peers, and less and less entries about more remote peers. These tables are populated when the peer joins the overlay, and are kept up to date by periodically updating them.

We recommend that, while using the full SHA-1 hash algorithm, peers maintain less than the full 160 entries in the finger table, perhaps 16 entries for small networks, 32 for larger networks. As this affects only the efficiency of the client, it is left to the implementer to determine a useful value. Note that a client can

easily store enough finger table entries to exceed the maximum MTU size when transmitting the full finger table. In this case, a client may need to reduce the number of finger table entries reported in DHT-Link headers.

6.2. Starting a New Overlay

A peer starting an overlay for the first time need not do anything special in order to construct the overlay. The peer MUST initialize its finger table. To create the finger table, a peer MUST take its Peer-ID and, by applying the exponential offsets for each finger, calculate the Resource-IDs corresponding to the start of each finger interval. See Finger Table, Successors and Predecessors (Section 6.1) for more details. After the finger table is created, the peer MUST initialize the finger table entries such that all entries point to itself. The peer MUST set its successor to itself, and MUST set its predecessor to NULL.

6.3. Peer Admission

A peer that wishes to join an overlay (called the joining peer), constructs a Peer Registration message and sends it to the bootstrap peer. The Peer Registration is routed to the admitting peer, which is the peer that is currently responsible for the joining peer's portion of the overlay.

6.3.1. Constructing a Peer Registration

To initiate the joining process, the joining peer constructs a Peer Registration and sends it to the bootstrap peer. The joining peer MUST construct the Peer Registration according the rules outlined in the dSIP [I-D.bryan-p2psip-dsip] draft. The joining peer MUST provide a DHT-PeerID header field in the Peer Registration and the dht-param part of the DHT-PeerID MUST be set to "*" or the token specified in the DHT Name Parameter (Section 5.1.2) section of this document.

Assume that a peer running on IP address 192.0.2.2 on port 5060 attempts to join the network by contacting a bootstrap peer at address 192.0.2.129. Further assume that 192.0.2.2 hashes to 463ac4b449 under SHA-1 (using a 10 digit hash for example simplicity), and that the overlay name is chat. An example message would look like this (neglecting tags):

```
REGISTER sip:192.0.2.129 SIP/2.0
To: <sip:peer@192.0.2.2;peer-ID=463ac4b449>
From: <sip:peer@192.0.2.2;peer-ID=463ac4b449>
Contact: <sip:peer@192.0.2.2;peer-ID=463ac4b449>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.2;peer-ID=463ac4b449>;algorithm=sha1;
            dht-param=Chord1.0;overlay=chat;expires=600
Require: dht
Supported: dht
```

6.3.2. Processing and Routing the Peer Registration

The Peer Registration is processed and routed according the rules outlined in the dSIP [I-D.bryan-p2psip-dsip] draft.

6.3.3. Admitting the Joining Peer

The admitting peer recognizes that it is presently responsible for this region of the hash space -- that is, it is currently the peer storing the information that the joining peer will eventually be responsible for. The admitting peer knows this because the joining peer's Peer-ID falls between the admitting peer's predecessor's Peer-ID and the admitting peer's Peer-ID. The admitting peer is responsible for helping the joining peer become a member of the overlay.

When handling a Peer Registration from a joining peer, the admitting peer MUST reply with a 200 response if the joining peer has a Peer-ID between the admitting peer's predecessor's Peer-ID and the admitting Peer-ID, or the admitting peer's predecessor is NULL. In the special case where the admitting peer's predecessor is NULL (as can happen if the admitting peer is the peer that started the overlay), that peer MUST reply with a 200 response to any peer validly attempting to join the system, regardless of Peer-ID.

The admitting peer MUST verify that the joining peer's Peer-ID is valid. If the joining peer's credentials are not valid, the message should be rejected with a response of 493 Undecipherable. In addition to verifying that the joining peer's Peer-ID is valid, the admitting peer MAY require an authentication challenge to the REGISTER message. Once any challenge has been met, the admitting will reply with a 200 OK message to the joining peer. As in a traditional registration, the Contact in the 200 OK will be the same as in the request, and the expiry time MUST be provided.

The 200 response that is constructed MUST provide information about the admitting peer's neighbors and finger table entries in the DHT-Link headers of the 200 response. This enables the joining peer to

initialize its neighbors and finger table entries. Additionally, the admitting peer MUST include its DHT-PeerID header containing the admitting peer's Peer-ID and IP. If the admitting peer's predecessor is not NULL, it MUST provide the joining peer with its current predecessor and successor in the 200. If the predecessor is NULL, the 200 MUST NOT include a value for the predecessor but MUST include a value for the successor. These MUST be placed placed in DHT-Link headers, as described in The DHT-Link Header (Section 5.2) section of this document. The predecessor MUST be transmitted in a DHT-Link header using a type of P and a depth of 1. The successor MUST be transmitted in a DHT-Link header using a type of S and a depth of 1.

The admitting peer SHOULD send a copy of the entries in their finger table to the joining peer, using DHT-Link headers of the F type. As the joining peer will likely be nearby the admitting peer in the hash space (at least for an overlay with a reasonable number of peers), this finger table information can likely improve the performance of the queries required to obtain a correct finger table information.

Continuing the example Peer Registration from the section above, assume now that the peer with Peer-ID 47e46fa2cd and IP address 192.0.2.7 is currently responsible for 463ac4b449 in the namespace. The admitting peer here does send the finger table, but we show only the first entry entry for clarity. We also omit the additional successors used to support redundancy for clarity. The response would look something like:

```
SIP/2.0 200 OK
To: <sip:peer@192.0.2.2;peer-ID=463ac4b449>
From: <sip:peer@192.0.2.2;peer-ID=463ac4b449>
Contact: <sip:peer@192.0.2.2;peer-ID=463ac4b449>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.2;peer-ID=463ac4b449>;algorithm=sha1;
            dht-param=Chord1.0;overlay=chat;expires=600
DHT-Link: <sip:peer@192.0.2.1;peer-ID=4201034a89>;link=P1;expires=412
DHT-Link: <sip:peer@192.0.2.227;peer-ID=574fb2d34a>;link=S1;expires=816
DHT-Link: <sip:peer@192.0.2.67;peer-ID=5f8dd34100>;link=F2;expires=121
Require: dht
Supported: dht
```

The joining peer obtains the Peer-ID and address of the admitting peer from the DHT-Peer header, and the information about the admitting peer's predecessor from the DHT-Link P 1 header. The joining peer MUST set its successor to be the admitting peer and its predecessor to be the admitting peer's predecessor. If the admitting peer did not provide a predecessor (which MUST only occur if the admitting peer's predecessor is NULL), the joining peer should leave their predecessor as NULL.

After the admitting peer sends the 200 response, it MUST set its predecessor to be the joining peer, and MUST obtain the information from the DHT-Peer header in the register request. It MUST NOT change the value of the predecessor prior to sending the 200. The admitting peer's successor is unchanged. Note that at this point the joining can also set all fingers pointing to intervals before the successor in the finger table to point to the successor.

Following the admission, the joining peer MUST run periodic stabilization as described in DHT Maintenance (Section 6.6). The admitting peer should perform periodic stabilization as well.

6.4. Chord Query Processing

A reply that is constructed to a query by the responsible peer MUST provide the current predecessor (if not NULL) and successor in the 200 or 404 message. These MUST be placed placed in DHT-Link headers, as described in The DHT-Link Header (Section 5.2) section of this document. If the predecessor is not NULL it MUST be transmitted in a DHT-Link header using a type of P and a depth of 1. It must be omitted if NULL. The successor MUST be transmitted in a DHT-Link header using a type of S and a depth of 1. The 200 or 404 SHOULD contain a copy of entries in their finger table, using DHT-Link headers with type F. Additionally, the replying peer MUST include its DHT-PeerID header.

6.5. Chord Graceful Leaving

Peers MUST send their resource registrations to their successor before leaving the overlay. Additionally, peers MUST unregister themselves with both their successor and predecessor. This unregister is constructed exactly the same as the Peer Registration message used to join, with the following exceptions. The expires parameter or header MUST be provided, and MUST be set to 0.

When a peer sends its unregister message to its successor and predecessor, it MUST include DHT-Link headers listing its predecessor and successor peers. This allows the peers receiving the requests to obtain the information needed to correct their predecessor and successor peers, as well as keep their successor lists needed for redundancy current.

OPEN ISSUE: should it be possible to trigger another peer to check its predecessor?

6.6. DHT Maintenance

In order to keep the overlay stable, peers must periodically perform book keeping operations to take into account peer failures. Periodically (we suggest 60-360 seconds), peers MUST perform finger table updates and periodic stabilization.

6.6.1. Chord Finger Table Updates

To update the finger table, a peer performs a Peer Query search for ID corresponding to the start of each of its finger intervals. These intervals are described in Finger Table, Successors and Predecessors (Section 6.1) and Peer Queries are described in the dSIP [I-D.bryan-p2psip-dsip] draft.

The 200 response to each Peer Query will contain the peer responsible for that ID in the DHT-PeerID header. The peer information in the DHT-PeerID header is entered into the corresponding finger table entry.

6.6.2. Chord Periodic Stabilization

In periodic stabilization, peers MUST perform an arbitrary query for their current successor's Peer-ID. The peer should examine the response from their successor. The predecessor reported should be the peer that made the request. If it is not, the peer MUST update their own successor with the predecessor returned, and additionally MUST send a REGISTER to this peer, structured as if the stabilizing peer had just entered the system. However, the peer sending this message MUST not process the response, but simply discard it, as this is intended only to pass information. This will serve to properly update the overlay. This is analogous to the notify procedure in Chord.

OPEN ISSUE: this operation is identical to the original chord operation, but it seems like we can pay attention to the response and observe if there have been multiple peers inserted and the peer we sent the REGISTER to knows a better successor peer, but this goes away with the next stabilize, anyway.

6.7. Peer Failure

Peer failure is handled by the periodic stabilization and responses to failed requests discussed above. Redundancy prevents against lost registrations.

6.8. Resource Replicas

When a resource is registered, the registering peer SHOULD create at least 2 redundant replicas to ensure the registry information is secure in the DHT. The registering peer is responsible for maintaining these replicas along with the primary entry.

7. Examples

For our examples, we use a simplified network. Rather than use a full SHA-1 hash, and the resulting 2^160 namespace, we instead use a smaller 4 bit hash, leading to a namespace of size 16. All hash results in our examples are contrived. We list the Peer-ID and Resource-IDs as xx, where xx is a number between 0 and 15 (2^4 namespace). In a real situation, the full 40 hex chars would be used. Additionally, because the number of finger table entries is so small in this case, we use the full 4 entries, where in a real case we suggest that one uses less than the number of bits in the namespace.

The empty overlay can be visualized as a circle with 16 possible vacant points, each corresponding to one possible location in the hash space. On the left, we have labeled these locations in the hash space as 0-15, starting in the upper left, and have used 0s to indicate vacant spaces in the hash space. On the right, we show the same network with 3 operating peers, denoted by capital Ns, with Peer-IDs of 3, 5, and 10. We will use this sample network state as the starting point for all our networks:

```
           0     1     2                  0     1     2
            0----0----0                    0----0----0
           /           \                  /           \
       15 0             0 3           15 0             N 3
         /               \              /               \
     14 0                 0 4       14 0                 0 4
        |                 |            |                 |
     13 0                 0 5       13 0                 N 5
        |                 |            |                 |
     12 0                 0 6       12 0                 0 6
         \               /              \               /
       11 0             0 7           11 0             0 7
           \           /                  \           /
            0----0----0                    N----0----0
          10     9     8                 10     9     8
```

Further, for the sake of example simplicity, assume the peer Peer-ID 3 has IP address 192.0.2.3, the peer peer with Peer-ID 5 has IP address 192.0.2.5, etc.

Data that hashes to a Resource-ID is stored by the next peer whose Peer-ID is equal to or larger than the Resource-ID, mod the size of the hash. As such, Peer 3 is responsible for any resources hashing from 11-15, as well as 0-3. Peer 5 is responsible for resources with Resource-IDs from 4-5, and Peer 10 is responsible for resources with Resource-IDs from 6-10. From this illustration, you follow a location clockwise until you encounter a peer, and this is the peer responsible for storing the information. This is illustrated below:

```
            0     1     2
             0----0----0
            /           \
        15 0             N 3
          /
      14 0                 0 4
         |                 |
      13 0                 N 5
         |
      12 0                 0 6
          \               /
        11 0             0 7
                        /
             N----0----0
           10     9     8
```

Finger tables give pointers to nearby peers. For our system, with 4 bit identifiers, we have 4 finger table entries. These finger tables point to the peer nearest to Peer-ID + 2^0, Peer-ID + 2^1, Peer-ID + 2^2 and Peer-ID + 2^3. If no peer is present at that location, the next available peer will be used. Thus, for our 3 peers, the finger tables look like the following, with ranges (indicated in traditional mathematical form) mapping to the peer those requests will be sent to:

	Peer 3	Peer 5	Peer 10
2^0 Entry	[4,5) -> 5	[6,7) -> 10	[11,12) -> 3
2^1 Entry	[5,7) -> 5	[7,9) -> 10	[12,14) -> 3
2^2 Entry	[7,11) -> 10	[9,13) -> 10	[14,2) -> 3
2^3 Entry	[11,3) -> 3	[13,5) -> 3	[2,10) -> 3

Assume further our sample network is called sipchat, and that 2 users are currently registered. User alice has a Resource-ID of 5, so her registration information is stored at peer 5. User bob is also registered, and has a Resource-ID of 12, so his registration information is stored by peer 3. Assume further that bob's UA is co-located with Peer 10, so his contact is sipchat/bob@192.0.2.10, and that alice is running a UA on a completely separate IP of 192.0.2.99, but is using an adapter peer running on Peer 3, therefore Peer 3 will send messages on alice's behalf, but alice's contact is sipchat/alice@192.0.2.99.

In each of the examples below, we assume we start from the network described above. Changes to the example network from previous examples are discarded.

Note that for simplicity we do not show user registration redundancy in any examples. This includes responses -- we only send predecessor and successor, as well as finger table -- not redundant successors.

7.1. Example of a Peer Registration

Assume a new peer wishes to join the system. The peer has an IP address of 192.0.2.14, which we shall assume hashes to a Peer-ID of 14. From an out of band mechanism, this peer discovers peer 5. This peer constructs a REGISTER and sends it to peer 5. Peer 5 verifies that 192.0.2.14 hashes to 14, then checks to see if it controls that portion of the namespace. Since it does not, it looks up in its finger table where it would route a search for 14, and determines it would send it to peer 3. The peer then sends a 302 back to peer 14, with a contact of peer 3.

Peer 14 the constructs a new REGISTER and sends it to Peer 3. Again, Peer 3 verifies the hash, and determines it is currently responsible for 14 in the hash space. After an optional challenge, it replies with a 200 OK message to admit the peer to the system. Finally, Peer 3 sends a third party registration on behalf of bob to Peer 14, transferring bob's registration to the new peer.

```
Peer 14             Peer 5              Peer 3
   |                   |                   |
   |(1) REGISTER       |                   |
   |------------------>|                   |
   |                   |                   |
   |(2) 302            |                   |
   |<------------------|                   |
   |                   |                   |
   |(3) REGISTER       |                   |
```

```
      |-------------------------------------->|
      |                   |                   |
      |(4) 200            |                   |
      |<--------------------------------------|
      |                   |                   |
      |(5) REGISTER       |                   |
      |<--------------------------------------|
      |                   |                   |
      |(6) 200            |                   |
      |-------------------------------------->|
      |                   |                   |
```

Peer 14 -> Peer 5

```
REGISTER sip:192.0.2.5 SIP/2.0
To: <sip:peer@192.0.2.14;peer-ID=14>
From: <sip:peer@192.0.2.14;peer-ID=14>
Contact: <sip:peer@192.0.2.14;peer-ID=14>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.14;peer-ID=14>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
Require: dht
Supported: dht
```

Peer 5 -> Peer 14

```
SIP/2.0 302 Moved Temporarily
To: <sip:peer@192.0.2.14;peer-ID=14>
From: <sip:peer@192.0.2.14;peer-ID=14>
Contact: <sip:peer@192.0.2.3;peer-ID=3>
DHT-PeerID: <sip:peer@192.0.2.5;peer-ID=5>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=1200
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=P1;expires=427
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=S1;expires=387
Require: dht
Supported: dht
```

Peer 14 -> Peer 3

```
REGISTER sip:192.0.2.3 SIP/2.0
To: <sip:peer@192.0.2.14;peer-ID=14>
From: <sip:peer@192.0.2.14;peer-ID=14>
Contact: <sip:peer@192.0.2.14;peer-ID=14>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.14;peer-ID=14r>;algorithm=sha1;
```

```
                dht=Chord1.0;overlay=chat;expires=600
   Require: dht
   Supported: dht


   Peer 3 -> Peer 14

   SIP/2.0 200 OK
   To: <sip:peer@192.0.2.14;peer-ID=14>
   From: <sip:peer@192.0.2.14;peer-ID=14>
   Contact: <sip:peer@192.0.2.14;peer-ID=14>
   Expires: 600
   DHT-PeerID: <sip:peer@192.0.2.3;peer-ID=3>;algorithm=sha1;
                 dht=Chord1.0;overlay=chat;expires=600
   DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=P1;expires=125
   DHT-Link: <sip:peer@192.0.2.5;peer-ID=5>;link=S1;expires=919
   DHT-Link: <sip:peer@192.0.2.5;peer-ID=5>;link=F0;expires=919
   DHT-Link: <sip:peer@192.0.2.5;peer-ID=5>;link=F1;expires=919
   DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=F2;expires=125
   DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F3;expires=600
   Require: dht
   Supported: dht


   Peer 3 -> Peer 14

   REGISTER sip:192.0.2.14 SIP/2.0
   To: <sip:bob@p2psip.org;resource-ID=12>
   From: <sip:peer@192.0.2.3;peer-ID=3>
   Contact: <sip:bob@192.0.2.10>
   Expires: 201
   DHT-PeerID: <sip:peer@192.0.2.3;peer-ID=3>;algorithm=sha1;
                dht=Chord1.0;overlay=chat;expires=600
   Require: dht
   Supported: dht


   Peer 14 -> Peer 3

   SIP/2.0 200 OK
   To: <sip:bob@p2psip.org;resource-ID=12>
   From: <sip:peer@192.0.2.3;peer-ID=3>
   Contact: <sip:bob@192.0.2.10>
   Expires: 201
   DHT-PeerID: <sip:peer@192.0.2.14;peer-ID=14>;algorithm=sha1;
                dht=Chord1.0;overlay=chat;expires=600
   Require: dht
   Supported: dht
```

7.2. Example of a User Registration

Assume user Carl starts a UA co-located with peer 5. Carl's contact will be carl@192.0.2.5, and his user name will be carl@p2psip.org. Carl's Peer hashes his user id and determines that the corresponding Resource-ID will be 11 -- that is, Carl's registration will be stored by the peer responsible for Resource-ID 11 -- ultimately Peer 3 in our example.

Carl's UA begins by constructing a SIP REGISTER message. Carl's UA consults its finger table, and determines that it should route requests pertaining to a Resource-ID of 11 to Peer 10. The REGISTER is sent to Peer 10, which observes that it is not responsible for that portion of the namespace, and consults the finger table, finding Peer 3 in the appropriate entry. Peer 10 sends a 302 containing Peer 3 as a contact.

Peer 5 constructs a new REGISTER on behalf of carl, and sends it to Peer 3. Peer 3 recognizes that it is responsible for storing this registration, and replies with a 200 OK (although in reality it might challenge in some way). The 200 contains some number of successor peers -- in this case 2 (although in our contrived example, one is peer 5 itself) that Carl's peer could send redundant registrations to. In our example, we do not show these. The 200 also (like 302s) must contain successors/predecessors in case the request is being used for stabilization. Again, in the tiny contrived example it looks odd since the second successor is the same as the predecessor. In a larger example this would not be the case.

[To Do: Maybe use a bigger example to fix these problems? That might be to big and ugly. Need a good way to show this]

```
  Peer 5              Peer 10             Peer 3
     |                   |                   |
     |(1) REGISTER       |                   |
     |------------------>|                   |
     |                   |                   |
     |(2) 302            |                   |
     |<------------------|                   |
     |                   |                   |
     |(3) REGISTER       |                   |
     |-------------------------------------->|
     |                   |                   |
     |(4) 200            |                   |
     |<--------------------------------------|
     |                   |                   |
```

```
Peer 5 -> Peer 10

REGISTER sip:192.0.2.10 SIP/2.0
To: <sip:carl@p2psip.org;resource-ID=11>
From: <sip:carl@p2psip.org;resource-ID=11>
Contact: <sip:carl@192.0.2.5>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.5;peer-ID=5>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=1200
Require: dht
Supported: dht


Peer 10 -> Peer 5

SIP/2.0 302 Moved Temporarily
Contact: <sip:peer@192.0.2.3;peer-ID=3>
DHT-PeerID: <sip:peer@192.0.2.10;peer-ID=10>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=800
DHT-Link: <sip:peer@192.0.2.5;peer-ID=5>;link=P1;expires=1200
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=S1;expires=412
Require: dht
Supported: dht


Peer 5 -> Peer 3

REGISTER sip:192.0.2.3 SIP/2.0
To: <sip:carl@p2psip.org;resource-ID=11>
From: <sip:carl@p2psip.org;resource-ID=11>
Contact: <sip:carl@192.0.2.5>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.5;peer-ID=5>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=1200
Require: dht
Supported: dht


Peer 3 -> Peer 5

SIP/2.0 200 OK
To: <sip:carl@p2psip.org;resource-ID=11>
From: <sip:carl@p2psip.org;resource-ID=11>
Contact: <sip:carl@192.0.2.5>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.3;peer-ID=3>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=P1;expires=405
```

```
DHT-Link: <sip:peer@192.0.2.5;peer-ID=5>;link=S1;expires=1200
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=S2;expires=405
Require: dht
Supported: dht
```

7.3. Example of a Session Establishment

Assume user Bob wishes to call user Alice. Bob's peer hashes Alice's user id, resulting in a Resource-ID of 5. Bob's peer (recall that Bob's UA is co-located with peer 10) consults its finger table, and determines that a request for Resource-ID 5 should be routed to Peer 3. A REGISTER query message is constructed and routed to Peer 3. Peer 3 determines it is not responsible for a Resource-ID of 5, looks up the ID in its finger table and determines it should be routed to Peer 5, so it returns a 302 referring to Peer 5. Bob's peer resends the REGISTER to Peer 5, which stores Alice's information. It sends a 200 with Alice's contact -- sipchat/alice@192.0.2.99. Bob finally sends an INVITE to Alice's UA, and session establishment is completed as normal.

```
  Peer 10             Peer 3             Peer 5           Alice UA
      |                   |                 |                  |
      |(1) REGISTER       |                 |                  |
      |------------------>|                 |                  |
      |                   |                 |                  |
      |(2) 302            |                 |                  |
      |<------------------|                 |                  |
      |                   |                 |                  |
      |(3) REGISTER       |                 |                  |
      |------------------------------------>|                  |
      |                   |                 |                  |
      |(4) 200            |                 |                  |
      |<------------------------------------|                  |
      |                   |                 |                  |
      |(5) INVITE         |                 |                  |
      |------------------------------------------------------->|
      |                   |                 |                  |
      |(6) 180            |                 |                  |
      |<-------------------------------------------------------|
      |                   |                 |                  |
      |(7) 200            |                 |                  |
      |<-------------------------------------------------------|
      |                   |                 |                  |
      |(8) ACK            |                 |                  |
      |------------------------------------------------------->|
```

```
    |                   |                |                  |


Peer 10 -> Peer 3

REGISTER sip:192.0.2.3 SIP/2.0
To: <sip:alice@p2psip.org;resource-ID=5>
From: <sip:bob@p2psip.org;resource-ID=12>
DHT-PeerID: <sip:peer@192.0.2.10;peer-ID=10>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=800
Require: dht
Supported: dht


Peer 3 -> Peer 10

SIP/2.0 302 Moved Temporarily
To: <sip:alice@p2psip.org;resource-ID=5>
From: <sip:bob@p2psip.org;resource-ID=12>
Contact: <sip:peer@192.0.2.5;peer-ID=5>
DHT-PeerID: <sip:peer@192.0.2.3;peer-ID=3>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=P1;expires=421
DHT-Link: <sip:peer@192.0.2.5;peer-ID=5>;link=S1;expires=1004
Require: dht
Supported: dht


Peer 10 -> Peer 5

REGISTER sip:192.0.2.5 SIP/2.0
To: <sip:alice@p2psip.org;resource-ID=5>
From: <sip:bob@p2psip.org;resource-ID=12>
DHT-PeerID: <sip:peer@192.0.2.10;peer-ID=10>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=800
Require: dht
Supported: dht


Peer 5 -> Peer 10

SIP/2.0 200 OK
To: <sip:alice@p2psip.org;resource-ID=5>
From: <sip:bob@p2psip.org;resource-ID=12>
Contact: <sip:alice@192.0.2.99>
DHT-PeerID: <sip:peer@192.0.2.5;peer-ID=5>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=1200
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=P1;expires=108
```

```
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=S1;expires=492
Require: dht
Supported: dht


Peer 10 -> Alice UA

INVITE sip:alice@p2psip.org SIP/2.0
To: <sip:alice@p2psip.org>
From: <sip:bob@p2psip.org>
Contact: <sip:bob@192.0.2.10>
DHT-PeerID: <sip:peer@192.0.2.10;peer-ID=10>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=800
Supported: dht
```

The remainder of the call is completed as any other SIP call. Note that if Alice's UA is DHT-compliant, then it will recognize the Supported field and DHT-PeerID header, and may respond with similar fields. However, if it does not support DHT extensions, it will simply ignore those values and complete the call as any normal non-P2P SIP UA.

7.4. Example of Moving From Empty Overlay to Stable 3 Peer System

In this example, we track the system state from overlay creation to a stable 3 peer overlay with 2 resources (user registrations) registered and stored in the system. This example will show how successor, predecessor, and finger table entries are updated during each step as well as show the P2P SIP messages exchanged.

Assume we start with an empty overlay with a namespace of 16. Each peer in the system will have one successor, one predecessor and 4 finger table entries (2^4 namespace). Peer state will be shown in the following way:

```
                    PeerID
     Successor
     Predecessor
     2^0 Entry
     2^1 Entry
     2^2 Entry
     2^3 Entry
     Resources

           0     1     2
            0----0----0
           /           \
       15 0             0 3
         /               \
     14 0                 0 4
        |                 |
     13 0                 0 5
        |                 |
     12 0                 0 6
         \               /
       11 0             0 7
           \           /
            0----0----0
          10     9     8
```

Additionally, we will track the location of resources with a resource map.

A peer with PeerID 3, IP address 192.0.2.3, and port 5060 starts the overlay called chat. When a peer starts a new overlay, it sets its successor to be its PeerID and sets its predecessor to be NULL. Additionally, the peer initializes its finger table and sets all fingers to be its PeerID.

The resulting state of the system is:

```
            0     1     2
             O----O----O
            /           \
        15 O             P 3
          /               \
      14 O                 O 4
         |                 |
      13 O                 O 5
         |                 |
      12 O                 O 6
          \               /
        11 O             O 7
            \           /
             O----O----O
           10     9     8

                    Peer 3
  Successor           3
  Predecessor        NULL
  2^0 Entry         [4,5)  -> 3
  2^1 Entry         [5,7)  -> 3
  2^2 Entry         [7,11) -> 3
  2^3 Entry         [11,3) -> 3
  Resources

Resource Map
```

For peer 3, there is a resource called alice that must be registered with the system. alice's peer hashes her user id and determines that the corresponding resource will be 8. alice's contact will be sipchat/alice@192.0.2.3 and her user name will be alice@p2psip.org. Because there are no other peers in the system, peer 3 is responsible for all resources including alice's registration. Note that we use "resID" as a short form for "resource-ID" to save room in the figure only -- resource-ID is used in the actual messages.

As a result the system state changes to:

```
          0     1     2
           0----0----0
          /           \
      15 0             P 3
        /               \
    14 0                 0 4
       |                 |
    13 0                 0 5
       |                 |
    12 0                 0 6
        \               /
      11 0             0 7
          \           /
           0----0----0
         10     9     8

                    Peer 3
    Successor           3
    Predecessor        NULL
    2^0 Entry         [4,5)  -> 3
    2^1 Entry         [5,7)  -> 3
    2^2 Entry         [7,11) -> 3
    2^3 Entry         [11,3) -> 3
    Resources         alice;resID=8 -> 3
```

```
Resource Map
   Resource name  ResID  ResLocation  ResStorage Location
     alice          8          3                3
```

Next a peer with PeerID 10, IP address 192.0.2.10, and port 5060 decides to join the overlay called chat. From an out of band mechanism, such as one of the ones listed in the Bootstrapping section of this document, this peer discovers peer 3. This new peer 10, constructs a REGISTER and sends it to peer 3.

Peer 3 verifies that 192.0.2.10 hashes to 10, then checks to see if it controls that portion of the namespace. Since Peer 3's predecessor is NULL and no other peers are in the system, Peer 3 determines that is currently responsible for peer 10 in the hash space. After an optional challenge, peer 3 replies with a 200 OK message to admit peer 10 into the system.

Peer 3 then sets its predecessor to be Peer 10. When Peer 10 receives the 200 OK message, it will set its successor to be Peer 3 and keeps its predecessor set to NULL (because no predecessor was in the 200 response).

Finally, Peer 3 sends a third party registration on behalf of alice

to Peer 10, transferring alice's registration to the new peer.

```
  Peer 10             Peer 3
     |                    |
     |(1) REGISTER        |
     |------------------->|
     |                    |
     |(2) 200             |
     |<-------------------|
     |                    |
     |(3) REGISTER        |
     |<-------------------|
     |                    |
     |(4) 200             |
     |------------------->|
     |                    |
```

Peer 10 -> Peer 3

```
REGISTER sip:192.0.2.3 SIP/2.0
To: <sip:peer@192.0.2.10;peer-ID=10>
From: <sip:peer@192.0.2.10;peer-ID=10>
Contact: <sip:peer@192.0.2.10;peer-ID=10>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.10;peer-ID=10>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
Require: dht
Supported: dht
```

Peer 3 -> Peer 10

```
SIP/2.0 200 OK
To: <sip:peer@192.0.2.10;peer-ID=10>
From: <sip:peer@192.0.2.10;peer-ID=10>
Contact: <sip:peer@192.0.2.10;peer-ID=10>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.3;peer-ID=3>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=S1;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F0;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F1;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F2;expires=125
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F3;expires=600
```

```
Require: dht
Supported: dht


Peer 3 -> Peer 10

REGISTER sip:192.0.2.10 SIP/2.0
To: <sip:alice@p2psip.org;resource-ID=8>
From: <sip:peer@192.0.2.3;peer-ID=3>
Contact: <sip:alice@192.0.2.3>
Expires: 201
DHT-PeerID: <sip:peer@192.0.2.3;peer-ID=3>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
Require: dht
Supported: dht


Peer 10 -> Peer 3

SIP/2.0 200 OK
To: <sip:alice@p2psip.org;resource-ID=8>
From: <sip:peer@192.0.2.3;peer-ID=3>
Contact: <sip:alice@192.0.2.3>
Expires: 201
DHT-PeerID: <sip:peer@192.0.2.10;peer-ID=10>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=S1;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F0;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F1;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F2;expires=125
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F3;expires=600
Require: dht
Supported: dht
```

The system state is now:

```
        0     1     2
         0----0----0
        /           \
    15 0             P 3
      /               \
  14 0                 0 4
     |                 |
  13 0                 0 5
     |                 |
  12 0                 0 6
      \               /
    11 0             0 7
        \           /
         P----0----0
       10     9     8

                    Peer 3                Peer 10
  Successor           3                     3
  Predecessor        10                    NULL
  2^0 Entry         [4,5)  -> 3           [11,12) -> 3
  2^1 Entry         [5,7)  -> 3           [12,14) -> 3
  2^2 Entry         [7,11) -> 3           [14, 2) -> 3
  2^3 Entry         [11,3) -> 3           [2, 10) -> 3
  Resources                               alice;resID=8 -> 3

Resource Map
   Resource name  ResID  ResLocation  ResStorage Location
     alice          8        3              10
```

Peer 10 runs its periodic stabilization. It constructs a REGISTER as described in the Peer Query section, and sends it to its successor, Peer 3, querying for its successor's PeerID.

Peer 3 checks the query to determine if it is responsible for the region the search key lies within. Because Peer 3's PeerID directly matches the search key, it sends a 200 OK response message with its current successor and predecessor specified in the DHT-Link headers.

Peer 10 examines the response from Peer 3. Because the predecessor in the response from Peer 3 is the same as Peer 10, the stabilizing peer, Peer 10 is not required to do any more work. Peer 10 should perform searches to update its finger table, but these are omitted for clarity.

```
  Peer 10             Peer 3
     |                   |
     |(1) REGISTER       |
     |------------------>|
     |                   |
     |(2) 200            |
     |<------------------|
```

Peer 10 -> Peer 3

```
REGISTER sip:192.0.2.3 SIP/2.0
To: <sip:peer@0.0.0.0;peer-ID=3>
From: <sip:peer@192.0.2.10;peer-ID=10>
DHT-PeerID: <sip:peer@192.0.2.10;peer-ID=10>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
Require: dht
Supported: dht
```

Peer 3 -> Peer 10

```
SIP/2.0 200 OK
To: <sip:peer@0.0.0.0;peer-ID=3>
From: <sip:peer@192.0.2.10;peer-ID=10>
Contact: <sip:peer@192.0.2.3;peer-ID=3>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.3;peer-ID=3>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=P1;expires=0
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=S1;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F0;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F1;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F2;expires=125
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F3;expires=600
Require: dht
Supported: dht
```

The state stays unchanged after Peer 10's periodic stabilization.

```
          0     1     2
           0----0----0
          /           \
      15 0             P 3
        /               \
    14 0                 0 4
       |                 |
    13 0                 0 5
       |                 |
    12 0                 0 6
        \               /
      11 0             0 7
          \           /
           P----0----0
         10     9     8

                    Peer 3               Peer 10
    Successor         3                    3
    Predecessor      10                   NULL
    2^0 Entry        [4,5)  -> 3          [11,12) -> 3
    2^1 Entry        [5,7)  -> 3          [12,14) -> 3
    2^2 Entry        [7,11) -> 3          [14, 2) -> 3
    2^3 Entry        [11,3) -> 3          [2, 10) -> 3
    Resources                             alice;resID=8 -> 3

   Resource Map
      Resource name  ResID  ResLocation  ResStorage Location
        alice          8        3               10
```

For peer 10, there is a resource called bob that must be registered with the system. bob's contact will be sipchat/bob@192.0.2.10 and his user name will be bob@p2psip.org. bob's peer hashes his user id and determines that the corresponding resource will be 11.

Bob's UA begins by constructing a SIP REGISTER message as described in Resource Registrations (Resource Registrations). Bob's UA consults its finger table, and determines that it should route requests pertaining to a Resource-ID of 5 to Peer 3. The REGISTER is sent to Peer 3, which observes that it is responsible for that portion of the namespace and replies with a 200 OK containing Peer 3's predecessor and successor information in the DHT-Link headers.

```
     Peer 10             Peer 3
        |                   |
        |(1) REGISTER       |
        |------------------>|
        |                   |
        |(2) 200            |
        |<------------------|
        |                   |
```

Peer 10 -> Peer 3

```
REGISTER sip:192.0.2.3 SIP/2.0
To: <sip:bob@p2psip.org;resource-ID=11>
From: <sip:bob@p2psip.org;resource-ID=11>
Contact: <sip:bob@192.0.2.10>
Expires: 231
DHT-PeerID: <sip:peer@192.0.2.10;peer-ID=10>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
Require: dht
Supported: dht
```

Peer 3 -> Peer 10

```
SIP/2.0 200 OK
To: <sip:bob@p2psip.org;resource-ID=11>
From: <sip:bob@p2psip.org;resource-ID=11>
Contact: <sip:bob@192.0.2.10>
Expires: 231
DHT-PeerID: <sip:peer@192.0.2.3;peer-ID=3>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=P1;expires=600
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=S1;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F0;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F1;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F2;expires=125
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F3;expires=600
Require: dht
Supported: dht
```

After bob's registration, the system state is:

```
           0     1     2
            0----0----0
           /           \
       15 0             P 3
         /               \
     14 0                 0 4
        |                 |
     13 0                 0 5
        |                 |
     12 0                 0 6
         \               /
       11 0             0 7
           \           /
            P----0----0
          10     9     8
```

	Peer 3	Peer 10
Successor	3	3
Predecessor	10	NULL
2^0 Entry	[4,5) -> 3	[11,12) -> 3
2^1 Entry	[5,7) -> 3	[12,14) -> 3
2^2 Entry	[7,11) -> 3	[14, 2) -> 3
2^3 Entry	[11,3) -> 3	[2, 10) -> 3
Resources	bob;resID=11 -> 10	alice;resID=8 -> 3

Resource Map

Resource name	ResID	ResLocation	ResStorage Location
alice	8	3	10
bob	11	10	3

Peer 3 now runs its periodic stabilization. It constructs a REGISTER as described in the Peer Query section, and sends it to its successor, Peer 3, querying for its successor's PeerID.

Peer 3 checks the query to determine if it is responsible for the region the search key lies within. Because Peer 3's PeerID directly matches the search key, it sends a 200 OK response message with its current successor and predecessor specified in the DHT-Link headers.

Peer 3 examines the response from itself. Because the predecessor in the response from Peer 3 is Peer 10, Peer 3 updates its successor to be Peer 10 and sends a REGISTER to Peer 10, structured as if Peer 3 had just entered the system.

When Peer 10 receives the message, it then sends a 200 OK response to Peer 3. Then, Peer 10 sets its predecessor to be Peer 3 because Peer 10's predecessor was NULL.

Then Peer 3 should perform searches to update its finger table, but these are simple peer queries and are omitted in this example. We show the finger table as though these searches were performed.

Note that because Peer 3's successor is Peer 3, we do not show such a REGISTER message being sent because implementations may choose to remove this step for efficiency. Rather we show the message sent from Peer 3 to Peer 10 that notifies Peer 10 that Peer 3 is its predecessor.

```
  Peer 3               Peer 10
     |                    |
     |(1) REGISTER        |
     |------------------->|
     |                    |
     |(2) 200             |
     |<-------------------|
     |                    |
```

Peer 3 -> Peer 10

```
REGISTER sip:192.0.2.10 SIP/2.0
To: <sip:peer@192.0.2.3;peer-ID=3>
From: <sip:peer@192.0.2.3;peer-ID=3>
Contact: <sip:peer@192.0.2.3;peer-ID=3>
Expires: 600
DHT-PeerID: <sip:3@3.0.0.3;peer-ID=>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
Require: dht
Supported: dht
```

Peer 10 -> Peer 3

```
SIP/2.0 200 OK
To: <sip:peer@192.0.2.3;peer-ID=3>
From: <sip:peer@192.0.2.3;peer-ID=3>
Contact: <sip:peer@192.0.2.3;peer-ID=3>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.10;peer-ID=10>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=S1;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F0;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F1;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F2;expires=125
```

```
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F3;expires=600
Require: dht
Supported: dht
```

After Peer 3's stabilization, the system state is:

```
        0     1     2
         0----0----0
        /           \
    15 0             P 3
      /               \
  14 0                 0 4
     |                 |
  13 0                 0 5
     |                 |
  12 0                 0 6
      \               /
    11 0             0 7
        \           /
         P----0----0
       10     9     8
```

	Peer 3	Peer 10
Successor	10	3
Predecessor	10	3
2^0 Entry	[4,5) -> 10	[11,12) -> 3
2^1 Entry	[5,7) -> 10	[12,14) -> 3
2^2 Entry	[7,11) -> 10	[14, 2) -> 3
2^3 Entry	[11,3) -> 3	[2, 10) -> 3
Resources	bob;resID=11 -> 10	alice;resID=8 -> 3

Resource Map

Resource name	ResID	ResLocation	ResStorage Location
alice	8	3	10
bob	11	10	3

Next a peer with PeerID 2, IP address 192.0.2.2, and port 5060 decides to join the overlay called chat. From an out of band mechanism, such as one of the ones listed in the Bootstrapping section of this document, this peer discovers peer 10. This new peer 2, constructs a REGISTER and sends it to peer 10.

Peer 10 verifies that 192.0.2.2 hashes to 2, then checks to see if it controls that portion of the namespace. Since it does not, it looks

up in its finger table where it would route a search for 2, and determines it would send it to peer 3. The peer then sends a 302 back to peer 2, with a contact of peer 3.

Peer 2 then constructs a new REGISTER and sends it to Peer 3. Again, Peer 3 verifies the hash, and determines it is currently responsible for 2 in the hash space. After an optional challenge, it replies with a 200 OK message to admit the peer to the system.

After sending the 200 response, Peer 3 then sets its predecessor to be Peer 2. When Peer 2 receives the 200 OK message, it will set its successor to be Peer 3 and set its predecessor to be Peer 10 (because that was the predecessor in the 200 response). Peer 2 should also perform searches to ensure that the finger table is up to date. These searches are omitted, but we update the finger table.

Finally, Peer 3 sends a third party registration on behalf of bob to Peer 2, transferring bob's registration to the new peer.

```
   Peer 2             Peer 10              Peer 3
     |                   |                   |
     |(1) REGISTER       |                   |
     |------------------>|                   |
     |                   |                   |
     |(2) 302            |                   |
     |<------------------|                   |
     |                   |                   |
     |(3) REGISTER       |                   |
     |-------------------------------------->|
     |                   |                   |
     |(4) 200            |                   |
     |<--------------------------------------|
     |                   |                   |
     |(5) REGISTER       |                   |
     |<--------------------------------------|
     |                   |                   |
     |(6) 200            |                   |
     |-------------------------------------->|
     |                   |                   |
```

Peer 2 -> Peer 10

```
REGISTER sip:192.0.2.10 SIP/2.0
To: <sip:peer@192.0.2.2;peer-ID=2>
From: <sip:peer@192.0.2.2;peer-ID=2>
```

```
Contact: <sip:peer@192.0.2.2;peer-ID=2>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.2;peer-ID=2>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
Require: dht
Supported: dht


Peer 10 -> Peer 2

SIP/2.0 302 Moved Temporarily
To: <sip:peer@192.0.2.2;peer-ID=2>
From: <sip:peer@192.0.2.2;peer-ID=2>
Contact: <sip:peer@192.0.2.3;peer-ID=3>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.10;peer-ID=10>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
DHT-Link: <sip:3@0.0.0.3;peer-ID=>;link=P1;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=S1;expires=919
Require: dht
Supported: dht


Peer 2 -> Peer 3

REGISTER sip:192.0.2.3 SIP/2.0
To: <sip:peer@192.0.2.2;peer-ID=2>
From: <sip:peer@192.0.2.2;peer-ID=2>
Contact: <sip:peer@192.0.2.2;peer-ID=2>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.2;peer-ID=2>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
Require: dht
Supported: dht


Peer 3 -> Peer 2

SIP/2.0 200 OK
To: <sip:peer@192.0.2.2;peer-ID=2>
From: <sip:peer@192.0.2.2;peer-ID=2>
Contact: <sip:peer@192.0.2.2;peer-ID=2>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.3;peer-ID=3>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=P1;expires=419
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=S1;expires=419
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=F0;expires=919
```

```
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=F1;expires=919
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=F2;expires=125
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F3;expires=600
Require: dht
Supported: dht


Peer 3 -> Peer 2

REGISTER sip:2.0.0.2 SIP/2.0
To: <sip:bob@p2psip.org;resource-ID=8>
From: <sip:peer@192.0.2.3;peer-ID=3>
Contact: <sip:bob@192.0.2.10>
Expires: 201
DHT-PeerID: <sip:peer@192.0.2.3;peer-ID=3>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
Require: dht
Supported: dht


Peer 2 -> Peer 3

SIP/2.0 200 OK
To: <sip:bob@p2psip.org;resource-ID=8>
From: <sip:peer@192.0.2.3;peer-ID=3>
Contact: <sip:bob@192.0.2.3>
Expires: 201
DHT-PeerID: <sip:peer@192.0.2.2;peer-ID=2>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=S1;expires=919
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>link=P1;expires=800
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F0;expires=919
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=F1;expires=919
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=F2;expires=125
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=F3;expires=600
Require: dht
Supported: dht
```

After Peer 2's insertion, the system state is:

```
         0     1     2
          0----0----P
         /           \
     15 0             P 3
       /               \
   14 0                 0 4
      |                 |
   13 0                 0 5
      |                 |
   12 0                 0 6
       \               /
     11 0             0 7
         \           /
          P----0----0
        10     9     8
```

	Peer 2	Peer 3	Peer 10
Successor	3	10	3
Predecessor	10	2	3
2^0 Entry	[3,4) -> 3	[4,5) -> 10	[11,12) -> 3
2^1 Entry	[4,6) -> 10	[5,7) -> 10	[12,14) -> 3
2^2 Entry	[6,10) -> 10	[7,11) -> 10	[14, 2) -> 3
2^3 Entry	[10,2) -> 10	[11, 3) -> 3	[2, 10) -> 3
Resources	bob;resID=11 -> 10		alice;resID=8 -> 3

Resource Map

Resource name	ResID	ResLocation	ResStorage Location
alice	8	3	10
bob	11	10	2

Peer 3 now runs its periodic stabilization. It constructs a REGISTER as described in the Peer Query section, and sends it to its successor, Peer 10, querying for its successor's PeerID.

Peer 10 checks the query to determine if it is responsible for the region the search key lies within. Because Peer 10's PeerID directly matches the search key, it sends a 200 OK response message with its current successor and predecessor specified in the DHT-Link headers.

Peer 3 examines the response from itself. Because the predecessor in the response form Peer 10 is the same as Peer3, the stabilizing peer, Peer 3 does not need to send any messages to the predecessor. At this point Peer 3 should send queries to ensure that the finger table is up to date. We do not show the SIP messages for this process, but do show the resulting changes in Peer 3's finger table.

```
    Peer 3              Peer 10
      |                    |
      |(1) REGISTER        |
      |------------------->|
      |                    |
      |(2) 200             |
      |<-------------------|
```

Peer 10 -> Peer 3

```
REGISTER sip:192.0.2.10 SIP/2.0
To: <sip:peer@0.0.0.0;peer-ID=10>
From: <sip:peer@192.0.2.3;peer-ID=3>
DHT-PeerID: <sip:peer@192.0.2.3;peer-ID=3>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
Require: dht
Supported: dht
```

Peer 10 -> Peer 3

```
SIP/2.0 200 OK
To: <sip:peer@0.0.0.0;peer-ID=10>
From: <sip:peer@192.0.2.3;peer-ID=3>
Contact: <sip:peer@192.0.2.10;peer-ID=10>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.10;peer-ID=10>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=P1;expires=0
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=S1;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F0;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F1;expires=919
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F2;expires=125
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F3;expires=600
Require: dht
Supported: dht
```

The state after Peer 3's periodic stabilization:

```
        0     1     2
         0----0----P
        /           \
    15 0             P 3
      /               \
  14 0                 0 4
     |                 |
  13 0                 0 5
     |                 |
  12 0                 0 6
      \               /
    11 0             0 7
        \           /
         P----0----0
       10     9     8
```

	Peer 2	Peer 3	Peer 10
Successor	3	10	3
Predecessor	10	2	3
2^0 Entry	[3,4) -> 3	[4,5) -> 10	[11,12) -> 3
2^1 Entry	[4,6) -> 10	[5,7) -> 10	[12,14) -> 3
2^2 Entry	[6,10) -> 10	[7,11) -> 10	[14, 2) -> 3
2^3 Entry	[10,2) -> 10	[11, 3) -> 2	[2, 10) -> 3
Resources	bob;resID=11->10		alice;resID=8 -> 3

Resource Map

Resource name	ResID	ResLocation	ResStorage Location
alice	8	3	10
bob	11	10	2

Peer 10 now runs its periodic stabilization. It constructs a REGISTER as described in the Peer Query section, and sends it to its successor, Peer 3, querying for its successor's PeerID.

Peer 3 checks the query to determine if it is responsible for the region the search key lies within. Because Peer 3's PeerID directly matches the search key, it sends a 200 OK response with its current successor and predecessor in the response.

Because the predecessor in the response from Peer 3 is Peer 2, Peer 10 updates its successor to be Peer 2 and sends a REGISTER to Peer 2, structured as if Peer 10 had just entered the system.

When Peer 2 receives the message, it then sends a 200 OK response to Peer 10. Then, Peer 2 updates its predecessor to be Peer 10.

Then Peer 10 should perform searches to update its finger table.

These are simple peer queries and are omitted in this example, but we have updated the finger table.

```
   Peer 10             Peer 3              Peer 2
      |                   |                    |
      |(1) REGISTER       |                    |
      |------------------>|                    |
      |                   |                    |
      |(2) 200            |                    |
      |<------------------|                    |
      |                   |                    |
      |(3) REGISTER       |                    |
      |--------------------------------------->|
      |                   |                    |
      |(4) 200            |                    |
      |<---------------------------------------|
      |                   |                    |
```

Peer 10 -> Peer 3

```
REGISTER sip:3.0.0.3 SIP/2.0
To: <sip:peer@0.0.0.0;peer-ID=3>
From: <sip:peer@192.0.2.10;peer-ID=10>
DHT-PeerID: <sip:peer@192.0.2.10;peer-ID=10>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
Require: dht
Supported: dht
```

Peer 3 -> Peer 10

```
SIP/2.0 200 OK
To: <sip:peer@0.0.0.0;peer-ID=3>
From: <sip:peer@192.0.2.10;peer-ID=10>
Contact: <sip:peer@192.0.2.3;peer-ID=3>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.3;peer-ID=3>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
DHT-Link: <sip:peer@192.0.2.2;peer-ID=2>;link=P1;expires=0
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=S1;expires=919
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=F0;expires=919
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=F1;expires=919
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=F2;expires=125
DHT-Link: <sip:peer@192.0.2.2;peer-ID=2>;link=F3;expires=600
Require: dht
Supported: dht
```

Peer 10 -> Peer 2

```
REGISTER sip:192.0.2.2 SIP/2.0
To: <sip:peer@192.0.2.10;peer-ID=10>
From: <sip:peer@192.0.2.10;peer-ID=10>
Contact: <sip:peer@192.0.2.10;peer-ID=10>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.10;peer-ID=10>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
Require: dht
Supported: dht
```

Peer 2 -> Peer 10

```
SIP/2.0 200 OK
To: <sip:peer@192.0.2.10;peer-ID=10>
From: <sip:peer@192.0.2.10;peer-ID=10>
Contact: <sip:peer@192.0.2.10;peer-ID=10>
Expires: 600
DHT-PeerID: <sip:peer@192.0.2.2;peer-ID=2>;algorithm=sha1;
            dht=Chord1.0;overlay=chat;expires=600
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=P1;expires=419
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=S1;expires=419
DHT-Link: <sip:peer@192.0.2.3;peer-ID=3>;link=F0;expires=919
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=F1;expires=919
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=F2;expires=125
DHT-Link: <sip:peer@192.0.2.10;peer-ID=10>;link=F3;expires=600
Require: dht
Supported: dht
```

After Peer 10's stabilization, the system state is:

```
           0     1     2
            O----O----P
           /          \
       15 O            P 3
         /              \
     14 O                O 4
        |                |
     13 O                O 5
        |                |
     12 O                O 6
         \              /
       11 O            O 7
           \          /
            P----O----O
          10     9     8
```

	Peer 2	Peer 3	Peer 10
Successor	3	10	2
Predecessor	10	2	3
2^0 Entry	[3,4) -> 3	[4,5) -> 10	[11,12) -> 3
2^1 Entry	[4,6) -> 10	[5,7) -> 10	[12,14) -> 3
2^2 Entry	[6,10) -> 10	[7,11) -> 10	[14, 2) -> 3
2^3 Entry	[10,2) -> 10	[11, 3) -> 2	[2, 10) -> 2
Resources	bob;resID=11 -> 10		alice;resID=8 -> 3

Resource Map

Resource name	ResID	ResLocation	ResStorage Location
alice	8	3	10
bob	11	10	2

This system now has 3 peers in a stable state, such that all predecessor and successor information is correct, and two user resources are stored in the overlay.

7.5. Example of a Peer Leaving the System

[To Do: Add an example here]

7.6. Example of a Successful User Search

[To Do: Add an example here]

7.7. Example of an Unsucessful User Search

[To Do: Add an example here]

8. Security Considerations

There are no new security considerations introduced in this draft beyond those already mentioned in the dSIP [I-D.bryan-p2psip-dsip] and Security for P2PSIP [I-D.lowekamp-p2psip-dsip-security] drafts.

9. Open Issues

There are certainly many open issues. Here are a few.

Should it be possible to trigger a node to recheck a finger table entry after it 302s to a node that appears to be down? Presumably this can be integrated together with the loose routing NAT traversal.

During graceful exit, should it be possible to trigger another peer to check its predecessor?

The periodic stabilization operation is identical to the original chord operation, but it seems like we can pay attention to the response and observe if there have been multiple peers inserted and the peer we sent the REGISTER to knows a better successor peer, but this goes away with the next stabilize, anyway.

10. Acknowledgements

A team of people have worked on the various drafts related to the dSIP protocol and extensions thereof. The team consists of: David Bryan, Eric Cooper, James Deverick, Cullen Jennings, Bruce Lowekamp, Philip Matthews, and Marcia Zangrilli.

Thanks to all who have been actively participating in the P2PSIP efforts. Special thanks to Spencer Dawkins, Enrico Marocco, and Jean-Francois Wauthy for providing editorial feedback, and Henry Sinnreich, Eric Rescorla, and Alan Johnston for various discussions related to this work.

11. IANA Considerations

This document defines the valid "link-value" values, and defines the "dht-param" value to be "Chord1.0".

12. Changes to this Version

While this is a -00 document, it has grown from sections of the earlier draft-bryan-sipping-p2p-xx.

13. References

13.1. Normative References

[I-D.bryan-p2psip-dsip]
Bryan, D., Lowekamp, B., and C. Jennings, "dSIP: A P2P Approach to SIP Registration and Resource Location", Internet Draft draft-bryan-p2psip-dsip-00, February 2007.

[I-D.lowekamp-p2psip-dsip-security]
Lowekamp, B. and J. Deverick, "Authenticated Identity Extensions to dSIP", Internet Draft draft-lowekamp-p2psip-dsip-security-00, February 2007.

[I-D.willis-p2psip-concepts]
Willis, D., "Concepts and Terminology for Peer to Peer SIP", draft-willis-p2psip-concepts-03 (work in progress), October 2006.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3174] Eastlake, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

13.2. Informative References

[Chord] Stoica, I., Morris, R., Liben-Nowell, D., Karger, D., Kaashoek, M., Dabek, F., and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", IEEE/ACM Transactions on Networking Volume 11, Issue 1, 17-32, Feb 2003.

Authors' Addresses

Marcia Zangrilli
SIPeerior Technologies
3000 Easter Circle
Williamsburg, VA 23188
USA

Phone: +1 757 565 0101
Email: marcia@sipeerior.com

David A. Bryan
SIPeerior Technologies
3000 Easter Circle
Williamsburg, VA 23188
USA

Phone: +1 757 565 0101
Email: dbryan@sipeerior.com

Full Copyright Statement

Intellectual Property

Acknowledgment

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