P2PSIP D. Bryan
Internet-Draft SIPeerior Technologies, Inc.
Intended status: Standards Track B. Lowekamp
Expires: August 29, 2007 SIPeerior; William & Mary
C. Jennings
Cisco Systems
February 25, 2007

dSIP: A P2P Approach to SIP Registration and Resource Location
draft-bryan-p2psip-dsip-00

Abstract

This document outlines the motivation, requirements, and architectural design for a distributed Session Initiation Protocol (dSIP). dSIP is a Peer-to-Peer (P2P) based approach for SIP registration and resource discovery using distributed hash tables maintained with SIP messages. This design removes the need for central servers from SIP, while offering full backward compatibility with SIP, allowing reuse of existing clients, and allowing P2P enabled peers to communicate with conventional SIP entities. A basic introduction to the concepts of P2P is presented, backward compatibility issues addressed, and security considerations are discussed.

dSIP is one possible implementation of the protocols being discussed for creation in the P2PSIP WG. In the context of the work being proposed, this draft represents a concrete proposal for the P2PSIP Peer Protocol, using SIP with extensions as the underlying protocol. In this architecture, no P2PSIP Client Protocol is needed, rather unmodified SIP is used for access by non-peers.

Table of Contents

1. Introduction

As SIP [1] and SIMPLE based Voice over IP (VoIP) and Instant Messaging (IM) systems have increased in popularity, situations have emerged where centralized servers are either inconvenient or undesirable. For example, a group of users wishing to communicate between each other, but using machines that are not consistently connected to the network, are often forced to use a central server that is outside the control of the group. Similarly, groups wishing to establish ephemeral networks for use in meetings, conferences, or classes often do not wish to configure a centralized server. Organizations may also want to allow their members to communicate with each other without traffic flowing to third parties, but may not have the staff or equipment to maintain a server.

Peer-to-Peer (P2P) computing has emerged as a mechanism for completely decentralized, server-free implementations of various applications. In particular, many recent efforts have focused on applying P2P to SIP within the IETF, starting with the forerunners of this document submitted by the authors. Since then a substantial usecases document [15] has emerged and, most recently, a concepts and terminology [2] document has helped define a common set of terms. This iteration of this document incorporates the terminology from that draft.

This draft presents dSIP, a SIP based system that uses P2P mechanisms to remove the need for central servers in SIP and SIMPLE based communications systems. While this draft evolved from early work done on the SoSIMPLE [16] P2PSIP project, it has changed extensively. This works reflects experience gained in actually building commercially available P2PSIP products based on this draft, as well as from extensive 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 [3].

Terminology defined in RFC 3261 [1] is used without definition.

We use the terminology and definitions from the Concepts and Terminology for Peer to Peer SIP [2] draft extensively in this document without further definition. Other terms used in this document are defined inline when used and are also defined below for reference.

In this illustrative purposes in this document we sometimes use 10 hexadecimal digit values for SHA-1 hashes. In reality, SHA-1 produces 40 digit values. They are shortened in this document for clarity and typographical considerations only.

2.1. Definitions

Peer-to-Peer (P2P) Architecture: An architecture in which peer nodes cooperate together to perform tasks. Each peer has essentially equal importance and performs the same tasks within the network. Additionally, peers communicate directly with one another to perform tasks. Contrast this to a Client-Server architecture.

Client-Server Architecture: An architecture in which some small number of nodes (servers) provide services to a larger number of nodes (clients). Client nodes initiate connections to servers, but typically do not communicate among themselves.

Conventional SIP: The architecture used by SIP as defined by RFC3261, RFC3263, and many others. Conventional SIP centralizes certain roles, such as registrar, but allows for direct end-to-end establishment of dialogs and media connections.

Distributed Hash Table (DHT): A mechanism in which resources are given a unique key produced by hashing some attribute of the resource, locating them in a hash space (see below). Peers located in this hash space also have a unique ID within the hash space. Peers store information about resources with keys that are numerically similar to the peer's ID in the hash space.

Namespace or hash space: The range of values that valid results from the hash algorithm fall into. For example, using the SHA-1 algorithm, the namespace is all 40 digit hexadecimal identifiers. This namespace forms the set of valid values for Peer-IDs and Resource-IDs (see below).

Routing Table: The list of peers that a peer uses to send messages to when routing. The structure and makeup of this table varies depending on the particular DHT selected.

Connection Table: A list of peers that the peer currently is maintaining open connections to. In general, this is a superset of the Routing Table. The extra entries may be cached entries for efficiency or additional entries needed for NAT traversal purposes.

Neighbors: A collection of peers that a particular peer can reach in one hop. In general, note that a peer's set of neighbors is equivalent to the entries in that peer's Routing Table. However, neighbors may include one or more peers that immediately precede the peer (predecessors) and one or more peers that immediately follow the peer in the namespace (successor peers). Note that neighbor relations do NOT have to be symmetric.

Adapter Peer: An adapter peer is a peer in the overlay that acts as an adapter for other non-P2P enabled SIP entities, allowing them to access the resources of the overlay. The adapter peer participates actively in the overlay network, while the non-P2P enabled SIP entities it provides service to DO NOT participate directly in the overlay. Compare these to the term "super peer" in the P2P community, although adapter peers may be thin software shims intended for only one client.

Peer Admission: The act of a peer joining the overlay. Registration allows a peer to communicate with other peers, and requires (allows?) it to take on some server-like responsibilities such as maintaining resource location information. It DOES NOT register the user so that they can receive phone calls, which is the conventional SIP use of the word registration. We refer to conventional SIP registration as "user registration".

User Registration: The act of a user registering themselves with a SIP network. User registration creates a mapping between a SIP URI and a contact for a user. This is the conventional meaning of registration in SIP. For a dSIP peer, this action MUST occur after peer registration. User or resource registration are terms used in this draft to refer to P2PSIP Resource Record Insertion, with the additional requirement that the resource's (user's) peer must first be admitted.

Joining Peer: During the peer admission process, this is the peer that is attempting to register -- that is, the peer that is attempting to join the overlay network.

Bootstrap Peer: During the process of peer registration, the bootstrap peer is the peer that the joining peer contacts. This peer may be a well-known peer, a peer located using a broadcast method, a peer that the joining peer previously knew about, or a peer that another bootstrap peer referred the joining peer to. The bootstrap peer MAY validate the joining peer's credentials and help the joining peer in opening connections to the admitting peer, but its primary purpose is to direct the joining peer to the admitting peer.

Admitting Peer: During the process of peer registration, this is the peer that is currently responsible for the portion of the namespace the new peer will eventually reside in. This peer is responsible for generating many of the messages exchanged during peer registration.

3. Background

3.1. Peer-to-Peer Fundamentals

The fundamental principle behind Peer-to-Peer (P2P) Architectures is that applications are provided by number of entities, called peers or

nodes working together with each other to accomplish tasks. Each and every peer is responsible for contributing to serving some of the transactions that take place on the network. Contrast this with the more traditional Client-Server Architecture in which a large number of clients communicate only with a small number of central servers responsible for performing tasks.

Each peer provides server-like functionality and services as well as being a client within the system. In this way, the services or resources that would be provided by a centralized entity are instead available in a distributed fashion from the peers of the system. Note that a particular peer may or may not provide a particular service, but some peer does, ensuring that collectively the peers can provide that particular service. The peers form a logical cluster of peers called an overlay or overlay network. The services provided are often said to be provided by the overlay, since collectively the members provide the services. The overlay is so named because they form a new, small sub-network at a higher logical level than lower level network connections.

In many P2P systems peers are assumed to be ephemeral in nature. A peer may join or leave the overlay at any time. The design of algorithms for P2P architectures take this into account. Information is replicated, and the topology of the overlay can be quickly adapted as peers enter and leave.

3.2. DHTs and Overlay Structure

While very early P2P systems used flood based techniques, most newer P2P systems locate resources using a Distributed Hash Table, or DHT to improve efficiency. Peers are organized using a Distributed Hash Table (DHT) structure. In such a system, every resource has a Resource-ID, which is obtained by hashing some keyword or value that uniquely identifies the resource. Resources can be thought of as being stored in a hash table at the entry corresponding to their Resource-ID. The peers that make up the overlay network are also assigned an ID, called a Peer-ID, in the same hash space as the Resource-IDs. A peer is responsible for storing all resources that have Resource-IDs near the peer's Peer-ID. The hash space is divided up so that all of the hash space is always the responsibility of some particular peer, although as peers enter and leave the system a particular peer's area may change. Messages are exchanged between the peers in the DHT as the peers enter and leave to preserve the structure of the DHT and exchange stored entries. Various DHT implementations may visualize the hash space as a grid, circle, or line.

Peers keep information about the location of other peers in the hash

space and typically know about many peers nearby in the hash space, and progressively fewer more distant peers. We refer to this table of other peers as a Routing Table. When a user wishes to search, they consult the list of peers they are aware of and contact the peer with the Peer-ID nearest the desired Resource-ID. If that peer does not know how to find the resource, it either returns information about a closer peer it knows about, or forwards the request to a closer peer. In this fashion, the request eventually reaches the peer responsible for the resource, which then replies to the requester.

3.3. P2PSIP

Unlike a conventional SIP architecture, P2PSIP systems require no central servers. In a conventional SIP architecture many UAs connect to one or more central servers which play a number of roles, including proxy server, registrar, presence server, and redirect server. In a P2PSIP architecture, the peers participating in the overlay not only act as conventional SIP UAs, allowing their users to place and receive calls, but, when viewed collectively with the other peers, perform the roles normally provided by a central server. Each participating peer will maintain some fraction of the information that would normally be maintained by the central servers in a conventional SIP network.

P2PSIP peers provide many functions, more than any single entity in a conventional SIP architecture. Minimally, a participating peer must be an active member of the overlay, participating in storage of resources, routing and providing some SIP "server-like" behaviors as well. In the terminology used in the concepts draft, these peers speak the P2PSIP Peer Protocol to organize among themselves.

The general concepts are more fully explained in the Concepts and Terminology for Peer to Peer SIP [2] draft.

4. The Architecture of dSIP

In this section we provide an overview of the architecture of dSIP and explain how it works in an informative way. Protocol details and syntax are provided in a normative form in the remainder of the document.

dSIP is a specific proposal for the P2PSIP Peer Protocol proposed in the Concepts and Terminology for Peer to Peer SIP [2] draft, using SIP messages as the syntax for encoding the protocol. The function of the P2PSIP Peer Protocol is to provide for mechanisms to maintain the overlay, as well as to store and retrieve information, and to

route messages when needed. dSIP's syntax is SIP with a number of newly defined headers, however no new methods are added to SIP in dSIP.

Because dSIP uses conventional SIP messages, the mechanisms used for NAT traversal in SIP, including STUN [4], TURN [17], and ICE [5] are reused, as explained in NAT Traversal for dSIP [6]. As a consequence, many peers are able to participate in the overlay even when behind NATs. For those that cannot for some reason, conventional SIP can be used, and these peers can connect using adapter peers, as described below. Since conventional SIP is used for this, there is no need for a P2PSIP Client Protocol, and therefore dSIP defines no such protocol.

dSIP is modular, allowing for the use of multiple DHTs, including those defined later. DHTs can be negotiated among the peers in much the same way as codecs or features are negotiated in conventional SIP. For compatibility, support for one basic DHT algorithm, Chord, is required. Additional DHTs can be added and supported. We detail the Chord algorithm for dSIP [7] and provide an alternate DHT algorithm for dSIP, based on Bamboo [8]. Note that this document does not specify the details of the DHTs, including Chord. These are defined in their own documents, which describe how the basic dSIP operations and syntax are used to implement that specific DHT algorithms. Our intention and hope is that others will design other overlay algorithms that rely on the same basic operations so that compatibility can be maintained.

4.1. Peer Functions and Behavior in dSIP

dSIP peers provide many functions, more than any single entity in a conventional SIP architecture. Minimally, a participating peer must be an active member of the overlay and must provide some SIP "server-like" behaviors as well. The code that implements the additional server-like and DHT behavior can be located in several places in the network. The simplest is to have peers that are endpoints directly joining the overlay as peers. In this case, these peers provide the basic functionality of any SIP endpoint, but additionally implement the operations described in this document to enable self-organization and provide SIP server-like functionality.

The behavior can also be located in an adapter peer, which allows one or more non-P2P aware SIP UAs (UAs that do not speak dSIP) to interact with the P2P overlay network. These adapters perform the additional self-organizing and SIP server-like behavior on behalf of the UA or UAs they support. In this case, only the adapter peer is a peer in the overlay, the UAs are not peers themselves. In this approach, the adaptors speak the P2PSIP Peer Protocol (dSIP in this

case), where the UAs speak conventional SIP. All interaction with the P2P overlay is carried out by the adapter peer. The adapter essentially acts as a proxy server for the unmodified SIP UAs. The adapter can take the form of a small software shim or may be code within a conventional RFC 3261 server.

In most places in this document, which type of peer we are discussing won't affect the discussion. In those cases where it will, we have noted the differences.

4.2. P2P Overlay Structure

The P2P overlay in dSIP consists of peers, which collectively serve as a directory service for locating resources (users, voicemail messages, etc.). Peers are organized using a supported Distributed Hash Table (DHT) P2P structure. dSIP allows for pluggable DHT algorithms the exact form of which is defined in the DHT algorithm definition.

Each peer is assigned a Peer-ID, and each resource that is stored in the overlay is assigned a Resource-ID. These values must map to the same name space. dSIP provides for various algorithms to be used to produce these values, although all members of the overlay must use the same algorithm. For example, in the Chord DHT implementation, SHA-1 is used to produce 160 bit values for both the Peer-ID and Resource-ID.

The Peer-ID assigned to each peer determines the peer's location in the DHT and the range of Resource-IDs for which it will be responsible. The exact mapping between these is determined by the DHT algorithm used by the overlay. The mechanism for selecting these Peer-IDs depends on the security mechanism being used by the overlay. For example, a simple SHA-1 hash of the IP address and the port of the peer could be used to generate the Peer-IDs or alternatively, a certificate based system in which CAs assign Peer-IDs could be used to obtain the Peer-IDs [9].

Every resource has a Resource-ID, obtained by hashing some keyword that identifies the resource. The Resource-IDs map to the same space as the Peer-IDs. In the case of users, the unique keyword is the userid and the resource is the registration -- a mapping between the user name and a contact. Resources can be thought of as being stored in the distributed hash table at a location corresponding to their Resource-ID. Because entities searching for resources must be able to locate them given the unique keyword, Resource-IDs are produced by hashing, and are never assigned, regardless of the DHT and security algorithms being used.

A resource with Resource-ID k will be stored by the peer with Peer-ID closest to the Resource-ID, as defined by the particular pluggable DHT algorithm being used. As peers enter and leave, resources may be stored on different peers, so the information related to them is exchanged as peers enter and leave. Redundancy is used to protect against loss of information in the event of a peer failure and to protect against compromised or subversive peers.

Since each DHT is defined and functions differently, we generically refer to the table of other peers that the DHT maintains and uses to route requests (neighbors) as a Routing Table. dSIP defines the syntax for the headers used to exchange these entries, but leaves the exact form of the data each DHT stores in the table as a decision for the DHT implementation. Peers may additionally maintain a list of peers to which they maintain connections for purposes other than routing, for example NAT traversal or caching. This larger table (usually a superset of the routing table) is referred to as the connection table in dSIP. In this draft, we refer to routing decisions being made from the entries in the routing table, although a peer might choose an entry from the connection table if it is a better match.

When locating a resource with a particular Resource-ID, the peer will send the request to the routing table entry with the Peer-ID closest to the desired Resource-ID, as defined by the particular DHT in use. Since DHTs must converge on the resource, the peer receiving the request is assumed to know of a peer with a Peer-ID closer to the Resource-ID, and responds by suggesting or forwarding the message to this peer, depending on the routing mechanism being used.

4.3. Use of SIP Messages in dSIP

dSIP uses SIP messages to implement the P2PSIP Peer Protocol. This was done for a number of reasons. In order to properly implement a P2PSIP protocol, it is necessary to have mechanisms to store, retrieve and query the locations of resources, as well as to route information. NAT traversal and security considerations require several techniques for routing information, as discussed below. Pluggable hashing techniques and DHT algorithms require capabilities to negotiate the use of these pluggable modules. We have found SIP offers mechanisms that meet all of these requirements today, has well defined security mechanisms, and additionally works well with the IETF suite of NAT traversal techniques: STUN, TURN and ICE. Because all this work would need to be redefined in a new P2PSIP protocol, and because all P2PSIP devices must, by definition, implement SIP anyway, we feel the only reasonable syntax choice for the P2PSIP Peer Protocol is SIP.

Our motivation throughout has been to preserve the semantics of conventional SIP messages to the extent possible. All of the messages that are needed to maintain the DHT, as well as those needed to query for information, are implemented using SIP messages. Fundamentally, messages are being exchanged for two purposes. The purpose of the first class of messages is to maintain the DHT, such as the messages needed to join or leave the overlay, and to transfer information between peers. The second type of message is the type most SIP users will be familiar with -- registering users, inviting other users to a session, etc. -- basic session establishment. As the DHT is used as a distributed registrar, the registration and other searches are performed within the DHT. Once the target resource has been located, further communication proceeds directly between the UAs (or designated adapter peers) as with conventional SIP communications.

The messages used to manipulate the DHT are SIP REGISTER messages. RFC 3261, Section 10.2, specifies that REGISTER messages are used to "add, remove, and query bindings." Accordingly, we have selected REGISTER methods to use to add, remove, and query bindings. We use REGISTER both for the bindings of hosts as neighbors (entries in the routing table) in DHT maintenance operations as well as the bindings of resource names to locations that are commonly maintained by SIP registrars. The only fundamental difference is that these operations occur within the overlay, rather than on the conventional server.

4.4. Routing in dSIP

When a peer sends a message within the DHT, it begins by calculating the target ID it is attempting to locate, using the particular algorithm used by the overlay. The target could be another user, a particular resource, or a peer (including itself) for DHT maintenance purposes. It then consults its routing table, and its other neighbor peers, for the closest peer it is aware of to the target ID, as defined by the closeness metric of the DHT in use.

In discussions of P2PSIP, several mechanisms have been discussed for routing. In each case, the initial message is sent from the requester to the peer in the routing table most likely to route correctly, as defined by the DHT algorithm in use. Subsequently, that peer may provide further routing using one of three mechanisms. These three types of routing are:

- o Iterative: If the contacted peer is not responsible for the target ID, then the contacted peer issues a 302 redirect response pointing the search peer toward the best match the contacted peer has for the target ID. The searching peer then contact the peer to which it has been redirected and the process iterates until the responsible peer is located.

- Recursive: If the contacted peer is not responsible for the target ID, it will forward the query to the nearest peer to the target that it knows, and the process repeats until the target is reached. The response unwinds and follows the same path on the message return. Because dSIP uses SIP messages for transport, SIP's proxy behavior is used to enable recursive routing.
- Semi-Recursive: Semi-Recursive is the same as Recursive routing on the outbound leg, but the reply "shortcuts" and is directly sent back to the requester. When discussing these techniques, we often just refer to Iterative and Recursive, because of the similarity between recursive and semi-recursive routing.

Various mechanisms may be used within the same overlay and even within the same search. For example, a search may start as iterative, but if a particular peer receiving the request knows that the requester cannot reach the next hop directly (perhaps due to NAT issues), the search may have recursive and iterative portions.

In general, the messages can be routed using any of these mechanisms, and this draft does not specify which mechanism will be used. The decision as to which mechanism is appropriate may be a factor of security, NAT traversal, or even the properties of the particular DHT being used. We generally refer to the message as being routed through the overlay.

4.4.1. dSIP Operations

dSIP provides mechanisms that are used for a number of operations.

4.4.1.1. Peer Registration

When a peer (called the joining peer) wishes to join the overlay, it determines its Peer-ID and sends a REGISTER message to a bootstrap peer already in the overlay, requesting to join. Any peer in the DHT may serve as a bootstrap peer. The mechanism for selecting bootstrap peers is application dependent, and discussed in Bootstrapping (Section 4.5).

Following the iterative routing scheme, the bootstrap peer looks up the peer it knows nearest to the Peer-ID of the joining peer and responds with 302 redirect to this closer peer. The joining peer will repeat this process until it reaches the peer currently responsible for the space it will occupy.

If recursive routing is being used, the bootstrap peer looks up the peer it knows nearest to the Peer-ID of the joining peer and forwards the REGISTER message to that peer. This process of forwarding the message repeats until the peer currently responsible for the space

the joining peer will occupy is found.

Once the peer responsible for the joining peer's portion of the namespace is located, the joining peer then exchanges DHT state information with this peer, called the admitting peer, to allow the joining peer to learn about other peers in the overlay (neighbors) and to obtain information about resources the joining peer will be responsible for maintaining. Other DHT maintenance messages will be exchanged later to maintain the overlay as other peers enter and leave, as well as to periodically verify the information about the overlay, but once the initial messages are exchanged, a peer has joined the overlay.

4.4.1.2. Resource Registration

The peer registration does not register the peer's user(s) or other resources with the P2PSIP network -- it has only allowed the peer to join the overlay. Once a peer has joined the overlay, the user that peer hosts must be registered with the system. This process is referred to as resource registration. This registration is analogous to the conventional SIP registration, in which a message is sent to the registrar creating a mapping between a SIP URI and a user's contact. The only difference is that since there is no central registrar, some peer in the overlay will maintain the registration on the users behalf.

Resource registrations are routed similarly to peer registrations. The resource's peer calculates the resource-ID and contacts the peer it is aware of nearest to the resource-ID. This search process continues in either an iterative or recursive manner until the responsible peer is located. This peer then stores the registration for that user and returns a 200 response.

For redundancy, resources should also be registered at additional peers within the overlay. These replicas are located by adding a replica number to the resource name and hashing to identify a new resource-ID for each replica. In this way, replicas are located at unrelated points around the DHT, minimizing the risk of an attacker compromising more than one registration for a single resource.

4.4.1.3. Session Establishment

Sessions are established by contacting the UA identified by the registration in the DHT. The first step in establishing a session is locating this peer, which is done by searching for a resource in the DHT. The name of the target resource is used to calculate a resource-ID and a REGISTER message with no Contact information (a conventional SIP search) is sent to the closest known peer to that

resource-ID. The search iterates until the responsible peer is located. The responsible peer then returns either a 200 OK with the Contact information for the resource or a 404 Not Found. The session is then initiated directly with the resource's UA.

4.4.1.4. DHT Maintenance

In order to keep the overlay stable, peers must periodically perform book keeping operation to take into account peer failures. These DHT maintenance messages are sent using REGISTER messages and the overlay algorithm being used will dictate how often and where these messages are sent.

DHT maintenance messages are routed similarly to peer registrations and resource registrations. The peer calculates the Peer-ID of the peer it wants to exchange DHT information with and contacts the peer it is aware of closest to that Peer-ID. This search process continues in either an iterative or recursive manner until the peer is located at which point the peers exchange DHT maintenance information.

4.5. Bootstrapping

When a peer wishes to join an existing overlay, it must first locate some peer that is already participating in the overlay, referred to as the bootstrap peer. Peers may use any method they choose to locate the initial bootstrap peer --- the decision is outside the scope of this specification. The following are a few of the many methods that may be used:

Static Locations: Some number of peers in the overlay may be persistent and have well know addresses. These addresses could be configured into the peer application or obtained using an out-of-band mechanism such as a web page.

Cached Peers: While this mechanism cannot be used the first time that a peer runs, on subsequent attempts to join the overlay a peer might attempt to use a previously contacted peer as a bootstrap peer.

Broadcast mechanisms: Peers can use a broadcast mechanism to locate the initial peer, for example by sending the first REGISTER message to the SIP multicast address.

5. Message Syntax

This section provides normative text explaining the syntax of the extensions we use for SIP messages.

5.1. Option Tags

We create a new option tag "dht" as described in RFC 3261. This option tag indicates support for DHT based P2PSIP. Peers MUST include a Require and Supported header with the option tag dht for all messages that are intended to be processed using dSIP or include P2P extensions. Clients supporting P2P and contacting another SIP entity using a non-P2P mechanism for a transaction that may or may later be P2P SHOULD include a Supported header with dht. For a typical session establishment the search within the DHT MUST specify Require dht, whereas the actual contact with the resource's UA SHOULD include a Supported header with dht but SHOULD NOT include a Require header with dht.

5.2. Hash Algorithms and Identifiers

All IDs used for an overlay must be calculated using the same algorithm. Implementations MUST support the SHA-1 algorithm, which produces a 160 bit hash value. The hash algorithm used is specified in the DHT-PeerID header, described below. An implementation MAY rely on a secret initialization vector, key, or other shared secret to use the identifier as an HMAC, from RFC 2104 [10] 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. See Security Extensions to the Distributed Session Initiation Protocol (dSIP) [9]for information on how to protect against replay attacks.

Both Peer-IDs and Resource-IDs MUST have the same range of values (map to the same space). Formally:

P2PID = token

When using SHA-1:

P2PID = 40LHEX

5.2.1. Peer-IDs

The particular DHT algorithm being used MAY specify an alternate mechanism for determining Peer-ID. Similarly, some security models may assign Peer-IDs from a central authority. In the event that neither of these mechanisms are being used, the Peer-ID MUST be formed by taking the IP address of the peer, without the colon or port, and with no leading zeros, and hashing this string with the hash algorithm. Then the least significant sixteen bits of the hash are replaced by the port used by the peer. For peers behind a NAT participating in an overlay on the public Internet, they must

identify their address on the public Internet through a protocol such as STUN [4] and use this address for their Peer-ID.

The string hashed to obtain the PeerID is formally defined below as ipaddress.

```
ipaddress         =  IPV4address / IPv6reference
```

PeerID is formally defined as:

```
PeerID = P2PID
```

5.2.2. Resource-IDs and the Replication

Resource-IDs MUST be formed by hashing the resource URI after converting it to canonical form. To do that, all URI parameters MUST be removed (including the user-param) except for the replica URI parameter, Any escaped characters MUST be converted to their unescaped form. Formally:

```
ResourceID = P2PID
```

5.3. P2PSIP URIs

Because hashing URIs to produce identifiers is a non-trivial cost, dSIP messages are constructed including these values already calculated. This is strictly as a courtesy to peers processing messages for this peer, as it prevents them from having to hash the URI again before routing. Identifiers provided in a message are a courtesy only and MUST NOT be used when making any changes to the data stored in an overlay, as they may be spoofed or incorrect. If the hash parameter is used incorrectly for routing, this only affects the transmitting peer's user. If it is used to insert or modify stored information, it can affect the system's integrity. Peers MUST verify the hash of all URIs before making changes that affect the overlay.

5.3.1. Peer URIs

A P2PSIP peer is represented by constructing a SIP-URI (or SIPS-URI) with the keyword "peer" or a short form of "P" for the userinfo portion. The URI parameter "peer-ID", or the short form "pID" MUST be used.

```
PeerURI        =   ("peer@" / "P@") hostport ";" PeerID-Param ";"
                   uri-parameters
```

Formally, the peerID uri-parameter is defined as type other-param

from RFC 3261 with a pname of "peerID" or "pID" for short form, and a pvalue which is of type PeerID. A peer receiving a PeerURI MUST verify the hash value of the PeerID-Param before using it to update its routing table.

```
PeerID-Param        =   ("peer-ID" / "pID") EQUAL PeerID
```

For search operations, where an identifier is being searched for, but the host responsible for that identifier is unknown, hostport MUST be set to "0.0.0.0". All non-search operations MUST specify a valid hostport.

P2P Peer URIs MUST NOT include the resource-ID URI parameter (below), as it is intended to define information about resources that are stored in the overlay, not information about the peers making up the overlay. P2P Peer URIs used in name-addr SHOULD NOT include any display-name information, and peers receiving name-addrs for peers with display-name information MUST ignore the information.

Examples, using a shortened hash for clarity:
The URI for a peer using the SHA-1 hash algorithm, with hashed ID ed57487add matching an IP address 10.6.5.5 used in a To header. Uses the short forms:

```
To: <sip:P@10.6.5.5;pID=ed57487add>
```

The URI for a peer using the SHA-1 hash algorithm, with hashed ID ed57487add matching an IP address 10.6.5.5 used in a To header. Uses the long forms:

```
To: <sip:peer@10.6.5.5;peer-ID=ed57487add>
```

5.3.2. Resource URIs and the resource-ID URI Parameter

Resource URIs are no different for P2PSIP resources than for non-P2P SIP applications. However, because calculating the ResourceID is a significant expense, the optional URI parameter resource-ID=<Resource-ID> or the short form rID=<Resource-ID> SHOULD be provided. This parameter is a courtesy only and MUST NOT be used when making any changes to the data stored in an overlay without being recalculated, as it may be spoofed or incorrect. The resource-ID URI parameter is of type other-param as defined in RFC 3261.

```
resourceID-param  =  ("resource-ID" \ "rID") EQUAL ResourceID
```

P2P Resource URIs MUST NOT include the PeerID-Param URI parameter, because this indicates that the target of the URI is a peer. P2P

Resource URIs MAY include other user-parameters such as user=phone.

Examples (again using shortened hashes for clarity):
The URI for a user resource with username bob@p2psip.org using the SHA-1 hash algorithm, with hashed Resource-ID 723fedaab1. The optional resource-ID URI parameter is included, using the long form:

```
sip:bob@p2psip.org;resource-ID=723fedaab1
```

The URI for a user resource with username bob@p2psip.org using the SHA-1 hash algorithm, with hashed Resource-ID 723fedaab1. The optional resource-ID URI parameter is included, using the short form:

```
sip:bob@p2psip.org;rID=723fedaab1
```

The URI, used in a To header for user Alice White, with username alice@p2psip.org. This example omits the optional resource-ID URI parameter:

```
To: "Alice White" <sip:alice@p2psip.org>
```

5.4. The DHT-PeerID Header and Overlay Parameters

We introduce a new SIP header called the DHT-PeerID header. This header is used to express the Peer-ID of the sending peer as well as to identify the name and parameters of the overlay. The format of the DHT-PeerID header is as follows:

```
DHT-PeerID  = "DHT-PeerID" HCOLON PeerURI SEMI algorithm SEMI
              dht-param SEMI overlay-param *(SEMI generic-param)
```

Examples:
A peer with an SHA-1 hashed Peer-ID of a04d371e on IP 192.168.1.1. We include the PeerURI, algorithm, dht-param, and overlay as well as the optional expires header parameter. In this example, the overlay name is chat and the DHT algorithm being used is dhtalg1.0

```
DHT-PeerID: <sip:peer@192.168.1.1;peer-ID=a04d371e>;algorithm=sha1;
            dht=dhtalg1.0;overlay=chat;expires=600
```

5.4.1. Hash Algorithms and the algorithm Parameter

The hash algorithm used for the overlay is specified as a parameter of the DHT-PeerID header. This parameter MUST appear in the DHT-PeerID header. It MUST be the algorithm used to calculate all PeerID

and ResourceID values used in the message. It SHOULD NOT appear in other headers in the message, but if it does it MUST match the value in the DHT-PeerID header.

The hash algorithm is specified using the algorithm parameter from RFC3261. The tokens used to identify the algorithm MUST be the same as those used in other SIP documents such as RFC4474. [11] Currently, those consist of 'sha1', indicating SHA-1 as defined in RFC 3174 [12] and 'hmac-sha1', indicating HMAC-SHA1 as defined in RFC2104 [10]. Implementations MUST support the SHA-1 algorithm.

A peer MUST reject a message with 488 Not Acceptable here if it specifies a different hash algorithm than that used by the peer's overlay. An initial contact to a bootstrap peer may specify the hash algorithm as the wildcard "*", in which case the joining peer indicates its willingness to use whatever hash algorithm the bootstrap peer identifies in its response. A peer responding to such a request MUST route the message according to the rules described in the Message Routing Section (Section 6) if all other elements of the message are correct and the routing algorithm indicates such a response is appropriate. If the normal response would be to allow the join with a 200 OK, the receiving peer MAY respond with a 302 redirect to itself and specifying the algorithm used in this overlay, in which case the joining peer should reissue the message with the proper hash algorithm specification.

5.4.2. Overlay Names and the overlay Parameter

Each overlay is named using a string, which SHOULD be unique to a particular deployment environment. Peers will use this value to identify messages in cases where they may belong to multiple overlays simultaneously. These are defined formally simply as a token:

```
overlay-name       = "*" / token
```

The overlay-param parameter MUST appear in the DHT-PeerID header. It SHOULD NOT appear in other headers in the message, but if it does it MUST match the value in the DHT-PeerID header. This parameter is defined formally as:

```
overlay-param      = "overlay" EQUAL overlay-name
```

A peer MUST reject a message with 488 Not Acceptable here if it specifies an overlay in which the peer is not participating. An initial contact to a bootstrap peer MAY specify overlay-name as the wildcard "*", in which case the joining peer indicates its willingness to join whatever overlay the bootstrap peer identifies in its response. A peer responding to such a request MUST route the

message according to the rules described in the Message Routing Section (Section 6) if all other elements of the message are correct and the routing algorithm indicates such a response is appropriate. If the normal response would be to allow the join with a 200 OK, the receiving peer MAY respond with a 302 redirect to itself, in which case the joining peer should reissue the message with the proper overlay specification.

5.4.3. DHT Algorithms and the dht Parameter

The routing algorithm used to implement the overlay is specified using a dht-param in the DHT-PeerID header. It SHOULD NOT appear in other headers in the message, but if it does it MUST match the value in the DHT-PeerID header. This parameter is defined formally as:

```
dht-name           = token
dht-param          = "dht" EQUAL dht-name
```

The behavior of a peer receiving a message with a dht-param specifying a routing algorithm other than that which it is following is dependent on the routing algorithm. An initial contact of a bootstrap peer MAY specify dht-param as the wildcard "*", in which case the joining peer indicates its willingness to use whatever DHT algorithm the bootstrap peer identifies in its response. A peer responding to such a request MUST route the message according to the rules described in the Message Routing Section (Section 6) if all other elements of the message are correct and the routing algorithm indicates such a response is appropriate. New routing algorithms SHOULD be designed to maintain backward compatibility with previous algorithms where possible. If the routing algorithm specified is incompatible, a 488 Not Acceptable Here response MUST be returned.

5.4.4. PeerID Expires header parameter

The DHT-PeerID header MAY include an Expires parameter indicating how long a recipient may keep knowledge of this peer. If not present, a default of 3600 is assumed. Mobile peers may wish to specify a shorter interval.

5.5. The DHT-Link Header

We introduce a new SIP header called 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 its neighbor peers and routing table information stored by a peer.

```
DHT-Link    = "DHT-Link" HCOLON PeerURI SEMI link-param SEMI
                   expires-param *(SEMI generic-param)
link-param      = "link" EQUAL link-value
expires-param   = "expires" EQUAL delta-seconds
```

The value of linkvalue -- that is, how you represent what type of link this is, is defined by the DHT algorithm specification. The generic-param leaves flexibility for an algorithm to add additional parameters if needed.

As an example, the header might look like (using a shortened 10 digit Peer-ID for clarity). The value *** here is intended to represent a value determined by the particular DHT:

```
DHT-Link: <sip:peer@192.168.0.1;Peer-ID=671a65bf22>;link=***;expires=600
```

5.5.1. Expires Processing

Each DHT-Link header MUST contain an expires parameter. Each peer maintains an expiration time for each of its neighbor and routing table entries. These expiration times are updated whenever the peer receives a response with a longer expiration time than it currently maintains, most commonly in the PeerID header of a response to a join or search. A peer MUST NOT report an expired entry in a DHT-Link header. A peer MUST update the expires parameter with the current value, adjusted for passed time, each time it generates a DHT-Link header.

6. Message Routing

When a peer sends a message within the DHT, it begins by calculating the target ID it is attempting to locate, which might be its own location in the DHT, or a user's registration, for which it hashes the user's URI to obtain the appropriate Resource-ID. It then consults its routing table, and its other neighbor peers, for the closest peer it is aware of to the target ID.

The messages in the overlay MAY be routed either iteratively or recursively. The Request-Disposition header as described in [13] SHOULD be used to indicate if the next node should process the message using a recursive or iterative mechanism. If the header is omitted, the receiving node may process the message either recursively or iteratively.

If the Request-Disposition header is iterative, the contacted peer MUST determine if it is responsible for that target ID. If it is not, then the contacted peer MUST issue a 302 redirect pointing the

search peer toward the best match the contacted peer has for the target ID. The searching peer then contact the peer to which it has been redirected and the process iterates until the responsible peer is located.

In recursive routing, the peer sends a message to the peer it knows that is nearest to the target. If the contacted peer is not responsible for the target ID, it MUST forward the query to the nearest peer to the target that it knows, and the process repeats until the target is reached. This process follows standard proxy behavior in RFC 3261.

6.1. Peer Registration

When a peer (the joining peer) wishes to join the overlay, it creates its Peer-ID and sends a REGISTER message to a bootstrap peer already in the overlay, requesting to join. Any peer in the DHT may serve as a bootstrap peer, although we expect that most UAs will be configured with a small number of well-known peers.

Following the iterative routing scheme, the bootstrap peer looks up the peer it knows nearest to the Peer-ID of the joining peer and responds with 302 redirect to this nearer peer. The joining peer will repeat this process until it reaches the peer currently responsible for the space it will occupy.

If recursive routing is being used, the bootstrap peer looks up the peer it knows nearest to the Peer-ID of the joining peer and forwards the REGISTER message to that peer. This process of forwarding the message repeats until the peer currently responsible for the space the joining peer will occupy is found.

Once the peer responsible for the joining peer's portion of the namespace is located, the joining peer then exchanges DHT state information with this peer, called the admitting peer, to allow the joining peer to learn about other peers in the overlay (neighbors) and to obtain information about resources the joining peer will be responsible for maintaining. Other DHT maintenance messages will be exchanged later to maintain the overlay as other peers enter and leave, as well as to periodically verify the information about the overlay, but once the initial messages are exchanged, a peer has joined the overlay.

6.2. Resource Registration

The peer registration does not register the peer's user(s) or other resources with the P2PSIP network -- it has only allowed the peer to join the overlay. Once a peer has joined the overlay, the user that

peer hosts must be registered with the system. This process is referred to as resource registration. This registration is analogous to the conventional SIP registration, in which a message is sent to the registrar creating a mapping between a SIP URI and a user's contact. The only difference is that since there is no central registrar, some peer in the overlay will maintain the registration on the users behalf.

Resource registrations are routed similarly to peer registrations. The resource's peer calculates the resource-ID and contacts the peer it is aware of closest to the resource-ID. This search process continues in either an iterative or recursive manner until the responsible peer is located. This peer then stores the registration for that user and returns a 200 response.

For redundancy, resources should also be registered at additional peers within the overlay. These replicas are located by adding a replica number to the resource name and hashing to identify a new resource-ID for each replica. In this way, replicas are located at unrelated points around the DHT, minimizing the risk of an attacker compromising more than one registration for a single resource.

6.3. Session Establishment

Sessions are established by contacting the UA identified by the registration in the DHT. The first step in establishing a session is locating this peer, which is done by searching for a resource in the DHT. The name of the target resource is used to calculate a resource-ID and a REGISTER message with no Contact information (a conventional SIP search) is sent to the closest known peer to that resource-ID. The search iterates until the responsible peer is located. The responsible peer then returns either a 200 OK with the Contact information for the resource or a 404 Not Found. The session is then initiated directly with the resource's UA.

If the peer needs to have the session establishment routed through the overlay, it MAY use the Request-Disposition header with a value of proxy to request that intermediate nodes proxy the invite over the overlay on their behalf. This is particular critical for NAT traversal [6].

6.4. DHT Maintenance

In order to keep the overlay stable, peers must periodically perform book keeping operations to take into account peer failures. These DHT maintenance messages are sent using REGISTER messages and the overlay algorithm being used will dictate how often and where these messages are sent.

DHT maintenance messages are routed similarly to peer registrations and resource registrations. The peer calculates the Peer-ID of the peer it wants to exchange DHT information with and contacts the peer it is aware of closest to that Peer-ID. This search process continues until the current closest peer to the target Peer-ID is located at which point the peers exchange DHT maintenance information.

7. Peer/DHT Operations

The SIP REGISTER message is used extensively in this system. REGISTER is used to register users, as in conventional SIP systems, and we discuss this further in the Resource Registration (Section 8.1) section of this document. Additionally, SIP REGISTER messages are used to register a new peer with the DHT and to transmit the information needed to maintain the DHT.

7.1. Peer Registration

After a peer has located an initial bootstrap peer, the process of joining the overlay is started by constructing a REGISTER message and sending it to the bootstrap peer. Third party registration MAY NOT be used for registering peers into the overlay, and attempts to do so MUST be rejected by the peer receiving such a request (although third party registrations are used for other purposes, as described below). The peer MUST construct a SIP REGISTER message following the instructions in RFC3261, Section 10, with the exceptions/rules outlined below.

7.1.1. Constructing a Peer Registration

The Request-URI MUST include only the IP address of the peer that is being contacted (initially the bootstrap peer). This URI MUST NOT include any of the P2P defined parameters. For example, a request intended for peer 10.3.44.2 should look like: "REGISTER sip:10.3.44.2 SIP/2.0".

The To and From fields of the REGISTER message MUST contain the URI of the registering peer constructed according to the rules in the subsection Peer URIs (Section 5.3.1) in the Message Syntax section.

While allowing the IP address of the sender for To and From is different than conventional SIP registers, there are two reasons for this. First, in a P2P network, which peer the request is sent to, and thus the domain for which the registration is intended, is not important. Any peer can process the information, and the user name is not associated with a particular IP address or DNS domain, but

rather with the overlay name, which is encoded elsewhere. In that sense, the IP address used is irrelevant. Choosing the domain of the sender ensures that if a request is sent to a non-P2P aware RFC 3261 compliant registrar, it will be rejected. RFC 3261 (section 10.3) states that a registrar should examine the To header to determine if it presents a valid address-of-record for the domain it serves. Since the IP address of the sending peer is unlikely to be a valid address for a non-P2P aware registrar, the message will be rejected, eliminating possibly erroneous handling by the registrar.

The registering peer MUST also list its PeerURI in the contact field when registering so that this may be identified as a registration/update, rather than a query. The peer MUST provide an expires parameter or expires header with a non-zero value. As in standard SIP registrations, Expire headers with a value of zero will be used to remove registrations.

The registering peer MUST provide a DHT-PeerID header field. It MAY leave the overlay parameter set to "*" for its initial registration message, but MUST set this parameter to the name of the overlay it is joining as soon as it receives a response from the bootstrap peer.

The registering peer MUST include Require and Supported headers with the option tag "dht".

Assume that a peer running on IP address 10.4.1.2 on port 5060 attempts to join the network by contacting a bootstrap peer at address 10.7.8.129. Further assume that 10.4.1.2 hashes to 463ac4b449 under SHA-1 (using a 10 digit hash for example simplicity), and the least significant bits are replaced with the port number, yielding 463ac413c4 and that the overlay name is chat and the dht-param is dhtalg1.0. An example message would look like this (neglecting tags):

```
REGISTER sip:10.7.8.129 SIP/2.0
To: <sip:peer@10.4.1.2;peer-ID=463ac413c4>
From: <sip:peer@10.4.1.2;peer-ID=463ac413c4>
Contact: <sip:peer@10.4.1.2;peer-ID=463ac413c4>
Expires: 600
DHT-PeerID: <sip:peer@10.4.1.2;peer-ID=463ac413c4>;algorithm=sha1;
            dht=dhtalg1.0;overlay=chat;expires=600
Require: dht
Supported: dht
```

7.1.2. Processing the Peer Registration

The receiving peer determines that this is a P2PSIP message based on the presence of the dht Require and Supported fields. In the event that the peer does not support P2P extensions, it MUST reply with a 420 Bad Extension response. If the peer examines the overlay parameters and determines that this is not an overlay the peer participates in, the peer MUST reject the message with a 488 Not Acceptable Here response. Likewise if the peer examines the dht-param and determines that the algorithm specified is not compatible with its algorithm, the peer MUST reject the message with a 488 Not Acceptable Here response. If a P2P peer receives a non-P2P request it MAY reject it with a message such as 421 Extension Required or it MAY process it as a conventional SIP message.

An implementation may support both P2P and conventional SIP messages. In that case, it MAY include the dht Supported field with all messages but MUST NOT include it with messages intended for conventional nodes.

7.1.2.1. Routing the Peer Registration

The presence of peer-ID URI parameter in the To and Contact headers and a valid expiration time indicate that this message is a peer registration and the receiving peer MUST process this as a DHT level request. The bootstrap peer SHOULD verify that the Peer-ID corresponds to peer listed in the URI by validating the hash or the peers credentials. If these do not match, the message SHOULD be rejected with a response of 493 Undecipherable. The bootstrap peer examines the Peer-ID to determine if it corresponds to the portion of the overlay the bootstrap peer is responsible for. If it does, the peer will handle the REGISTER request itself. If not, the bootstrap peer will either provide the joining peer with information about a peer closer to the area of the overlay where the joining peers Peer-ID is stored (iterative routing) or forward the request along the closest peer it knows about (recursive routing). If a Request-Disposition header is present and set to proxy, the peer MUST use a recursive routing mechanism, and if it is present and set to redirect, the peer MUST use an iterative routing mechanism. In the event that the Request-Disposition header is not present, the peer may choose either mechanism.

In the case of iterative routing, if the receiving peer is not responsible for the area of the hash table where Peer-ID should be stored, the peer SHOULD generate a 302 message. The 302 is constructed according the rules of RFC 3261 with the following rules. The receiving peer MUST look in its list of neighbors or in the routing table to find the peer with Peer-ID nearest the to joining

peer's Peer-ID, and use it to create a contact field in the form of a peer URI, as specified in the P2P Peer URIs (Section 5.3.1) section of this document, including appropriate URI parameters. The response MUST contain a valid DHT-PeerID header. This response is sent to the joining peer.

In the case of recursive routing, if the receiving peer is not responsible for the area of the hash table where the Peer-ID should be stored, the receiving peer should forward the request to the peer it knows about that is closest to the Peer-ID.

A peer MUST NOT add a new peer to its routing table or redirect requests to that new peer until it has successfully contacted that peer itself. By redirecting a message to another peer, the contacted peer indicates that it believes that peer to be alive and that it is willing to route messages to it for NAT and Firewall traversal purposes.

Using our example register from the previous section, assume that iterative routing is being used and that the bootstrap peer 10.7.8.129 receives the message, determines it is not responsible for that area of the overlay, and redirects the joining peer to a peer with Peer-ID 47e46fa2cd at IP address 10.3.1.7. The 302 response, again neglecting tags, is shown below. Note that the peer creating the response uses its information to construct the DHT-PeerID header.

```
SIP/2.0 302 Moved Temporarily
To: <sip:peer@10.4.1.2;peerId=463ac413c4>
From: <sip:peer@10.4.1.2;peerId=463ac413c4>
Contact: <sip:peer@10.3.1.7;peerId=47e46fa2cd>
Expires: 600
DHT-PeerID: <sip:@10.7.8.129;peerId=084d299ff2>;algorithm=sha1;
            dht-param=dhtalg1.0;overlay=chat;expires=600
Require: dht
Supported: dht
```

Upon receiving the 302, the joining peer uses the contact address as the new bootstrap peer. The process is repeated until the peer contacted is currently responsible for the area of the DHT in which the new peer will reside. The receiving peer that is responsible for that portion of the overlay is referred to as the admitting peer.

TODO: we should have example of how to forward request in a recursive routing case

7.1.2.2. Admitting the Joining Peer

The admitting peer MUST verify that the Peer-ID is valid, as described above. If these do not match, the message MUST be rejected with a response of 493 Undecipherable. 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 this Peer-Id will eventually be responsible for. The admitting peer knows this because the joining peer's Peer-ID is closest to its own Peer-ID. The admitting peer is responsible for helping the joining peer become a member of the overlay. In addition to verifying that the Peer-ID was properly calculated, the admitting peer MAY perform additional security checks [9]. Once any challenge has been met, the admitting will reply with a 200 OK message to the joining peer. As in a conventional registration, the Contact in the 200 OK will be the same as in the request, and the expiry time MUST be provided.

The admitting peer MUST reply with a 200 response if the admitting peer's Peer-ID is the closest to the joining peer's Peer-ID. Each DHT algorithm MAY choose to define closest however they want, but the DHT algorithm MUST be able to deterministically find the closest Peer-ID. The admitting peer must populate the DHT-Link headers with all values required by the DHT routing protocol so that the joining peer can initialize its neighbors and routing table entries. Additionally, the admitting peer MUST include its DHT-PeerID header containing the admitting peer's Peer-ID and IP.

7.2. Peer Query

As with conventional SIP, REGISTER messages that are sent without a Contact: header are assumed to be queries, as described in Section 10 of RFC3261.

7.2.1. Constructing a Peer Query Message

The peer looks for the routing table entry or neighbor peer that is closest to the ID they are searching for. If the routing table has not yet been filled, then the peer may send the request to any peer it has available, including their other neighbor peers or even some bootstrap peer. While these initial searches may be less efficient, they will succeed. The Request-URI MUST include only the IP address of the peer that the search is intended for. This URI MUST NOT include any of the P2P defined parameters. For example, a request intended for peer 10.3.44.2 should look like: "REGISTER sip:10.3.44.2 SIP/2.0".

Because this is a query, the sending peer MUST NOT include a contact header. The sender MUST NOT include an expires header.

The peer MUST provide a DHT-PeerID header.

The peer MUST include Require and Supported headers with the option tag "dht".

Assume that a peer running on IP address 10.4.1.2 on port 5060 wants to determine who is responsible for Peer-ID 4823affe45, and asks the peer with IP address 10.5.6.211 Further assume that the peer uses 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:10.5.6.211 SIP/2.0
To: <sip:peer@0.0.0.0;peerId=4823affe45>
From: <sip:peer@10.4.1.2;peerId=463ac413c4>
DHT-PeerID: <sip:peer@10.4.1.2;peerId=463av413c4>;algorithm=sha1;
            dht-param=dhtalg1.0;overlay=chat;expires=600
Require: dht
Supported: dht
```

The To field of the REGISTER message MUST contain the PeerURI of the identifier being search for, constructed according to the rules in the subsection P2P peer URIs (Section 5.3.1) in the Message Syntax section. If a specific peer is being sought, the PeerURI must specify that hostport. If only the identifier is being searched for, then hostport MUST be set to "0.0.0.0". The From URI MUST use the searching peer's PeerURI.

7.2.2. Processing Peer Query Message

The receiving peer determines that this is a P2PSIP message based on the presence of the dht Require and Supported fields. In the event that the peer does not support P2P extensions, it MUST reply with a 5xx class response such as 501 Not Implemented. If the peer examines the overlay parameters and determines that this is not an overlay the peer participates in, the peer MUST reject the message with a 488 Not Acceptable Here response. In the event a P2P peer receives a non-P2P request, it SHOULD reject it with a message such as 421 Extension Required.

7.2.2.1. Routing the Peer Query Message

The presence of a PeerURI and lack of an expiration time indicate that this message is a peer query and the receiving peer MUST process this as a DHT level request. The receiving peer SHOULD NOT alter any of its internal values such as successor or predecessor in response to this message, since it is a query. Otherwise, the message is processed and routed as a peer registration (Section 7.1.2.1) until

the responsible peer is reached.

7.2.2.2. Responding to the Peer Query Message

If the receiving peer is responsible for the region that the search key lies within, it MUST respond to the query. If the receiving peer's Peer-ID exactly matches the search key, it MUST respond with a 200 OK message. If it is responsible for that region, but its Peer-ID is not the search key, it MUST respond with a 404 Not Found message. The peer MAY verify the Peer-ID and IP address presented by the querying peer in the message. If these do not match, the message should be rejected with a response of 493 Undecipherable.

7.3. Populating the Joining Peer's Routing Table

Once admitted, the joining peer SHOULD populate its routing table and locate neighbors by issuing queries for peers with the appropriate identifiers. If the admitting peer provided neighbor or routing table information in its response, the joining peer MAY use this information to construct a temporary routing table and neighbor information and use this temporary table in the queries to populate the table.

7.4. Transfering User Registrations

When a new peer joins, it splits the area in the hash space the admitting peer is responsible for. Some portion of the user registrations the admitting peer was responsible for may now be the responsibility of the joining peer, and these user registrations are handed to the joining peer by means of third party user registrations. Third party registrations are allowed for user registrations and arbitrary searches, but are not allowed for peer registrations. These registrations are exactly the same as those discussed in Registering and Removing User Registrations (Section 8.1), except that as they are third party registration from a peer, that is, the From header contains the PeerURI of the admitting peer.

7.5. Peers Leaving the Overlay Gracefully

Peers MUST send their registrations to the closest peer before leaving the overlay, as described in the section above. Additionally, peers MUST unregister themselves with their symmetric neighbors (if the DHT routing algorithm uses symmetric neighbors in any form). These graceful exit REGISTER messages are constructed exactly the same as one used to join, with the following exceptions. The expires parameter or header MUST be provided, and MUST be set to 0. DHT-Link headers must be provided, as specified in DHT routing

algorithm

7.6. NAT and Firewall Traversal

The filtering properties of NATs and firewalls can lead to non-transitive connectivity. Typically this will manifest itself in a peer receiving a 302 redirecting it to another peer that it cannot contact, most likely because address dependent filtering is occurring. We discuss mechanisms to address these problems in [6].

7.7. Handling Failed Requests

When a request sent to another peer fails, the peer MUST perform searches to update its pointers. If the failed request was sent to a peer in the routing table or a neighbor peer, then the searches discussed in Populating the Joining Peer's Routing Table (Section 7.3) should be performed.

8. Resource Operations

The most important element of resource operations within the P2PSIP DHT is that they are performed exactly as if using a conventional SIP registrar, except that the registrar responsibilities are distributed among the DHT members.

8.1. Resource Registrations

When a peer is in the overlay, it must register the contacts for users and other resources for which it is responsible into the overlay. This differs from the registrations described above in that these registrations are responsible for entering a URI name to URI location mapping into the overlay as data, rather than joining a peer into the overlay. These registrations are very similar to those outlined in section 10 of RFC3261.

The Request-URI that is constructed for the REGISTER MUST be addressed to the peer the request is sent to. The To and From fields of the REGISTER message MUST contain the Resource URI of the resource being registered, as described in Resource URIs (Section 5.3.2). The request MUST include the value dht in Require and Supported headers. The request MUST include a DHT-PeerID header and MAY include one or more DHT-Link headers.

The resource registration MUST include at least one Contact header containing a location of the resource and allowing this to be identified as a registration/update, rather than a query. The peer MUST provide an expires parameter or an Expire header with a non-zero

value. As in standard SIP registrations, Expires parameters with a value of zero will be used to remove registrations. Any valid Contact for RFC 3261 is valid Contact for P2PSIP. Most users will register a Contact with the address of the user's UA (which may or may not be the IP address of the peer, since the peer could be an adaptor peer). The Contact URI does not need to include the ResourceID or other P2PSIP parameters as it is stored in the DHT but not processed or routed by it in any way.

The message is routed in a fashion exactly analogous to that described in the section on peer registration (Section 7.1). In iterative routing algorithms, 302 messages are sent to indicate that the message is to be redirected to another Peer URI. In recursive routing algorithms, the receiving peer SHOULD forward the request to the peer in its connection table that is closest to the ResourceID. Once the message arrives at a destination that is responsible for that portion of the hash namespace, the peer recognizes it as a resource registration, rather than a peer wishing to join the system, based upon the fact that the To and From fields do not contain a Peer URI. The peer responds with a 200 indicating a successful registration. The response is constructed as dictated by RFC3261.

The registering peer SHOULD construct and register replica registrations using the same Contact headers, but with the replica URI parameter used in the To and From headers.

8.2. Refreshing Resource Registrations

Resource registrations are refreshed exactly as described in RFC 3261, Section 10. Responsible peers should send a new registration with a valid expiration time prior to the time that the registration is set to expire.

Agents MAY cache the address where they previously registered and attempt to send refreshes to this peer, but they are not guaranteed success, as a new peer may have registered and may now be responsible for this area of the space. In such a case if iterative routing is being used, the peer will receive a 302 from the peer with which they previously registered, and should follow the same procedure for locating the peer they used in the initial registration.

As with initial registrations, the sending peer should use the neighbor peer or routing table information provided in the 200 to send these updates to the redundant peers as well.

8.3. Removing Resource Registrations

Resource registrations are removed exactly as described in RFC 3261, Section 10. Responsible peers MUST send a registration with expiration time of zero.

As with initial registrations, the sending peer MUST construct replica unregister messages and use these to unregister the replicas.

8.4. Querying Resource Registrations

Resource queries are constructed as described in RFC 3261, Section 10. Querying peers should send a REGISTER message with no contact header. As described in Peer Search (Section 7.2.1), this mechanism can also be used to locate the peer responsible for a particular Resource-ID.

A P2P environment can do little to protect against an individual peer compromising the registrations it is responsible for. Accordingly, a UA cannot trust a response from a single peer, whether it indicates a successful search or an error. In the absence of other methods of verifying the response (such as having a certificate of the user being searched for and a signed registration that can be verified with the certificate) a UA should search for the primary registration and at least one replica. Because the locations the replicas are stored are unrelated to the location of the primary registration, a single attacker is unlikely to be able to compromise both entries. As the overlay gains more peers and more replicas are searched for, the odds of a compromise are reduced. Better protection for registrations is discussed in [9].

8.5. Session Establishment

When a caller wishes to send a SIP message (such as an INVITE, MESSAGE or SUBSCRIBE), the caller must first locate the peer where this callee's information resides using the resource search procedure described in the section titled Resource Location. (Section 8.4)

Establishing a session is done entirely in the normal SIP fashion after the user is located using the P2P resource query. Once the peer responsible for the Resource-ID is located, it will provide either a 200, providing a contact for the users UA, or will provide a 404 if the user is not registered. If a 200 with a valid contact is received, the call will then be initiated directly with the UAS of the called using the standard RFC 3261 fashion for methods such as INVITE or MESSAGE, or the INVITE can be processed by routing it through the overlay if necessary for NAT traversal [6].

8.6. Presence

We use SUBSCRIBE/NOTIFY for this. We subscribe to every user on our friend list when we come online. If the friends are online, that means that we know exactly where they are. Peers MAY use the PeerIDs of their friends' peers as additional routing table entries or neighbor peers (essentially, cached values), consulting these first, as connections are likely to be made to people on the user's friend list. These should also be periodically checked, as described in the DHT Maintenance (Section 6.4).

If friends are offline, one should periodically try to make the connection. However, if a UA receives a SUBSCRIBE from a friend that it believes to be offline, it SHOULD attempt to subscribe to that friend. This will allow people that are reciprocally on each other's friend lists to rapidly be notified when one or the other comes online, therefore the retry interval for subscribing to offline friends can be fairly long because it is only necessary in the case of race conditions or other temporary failures in resource location.

8.7. Offline Storage

Delivery of messages to offline users, or voicemail for voice applications, requires storing that information for later retrieval. Storing user configuration information in a format accessible from the network also will allow a user to retrieve their profile from any computer. Cao et al. [18] describe an approach that separates the storage of resource location information from the actual storage of the offline research. We believe that this approach is in agreement with the approach taken by the rest of this document, which relies on the DHT overlay to store the registrar's location information, but relies on external, conventional methods for the actual connection. For offline storage, it also allows the use of other standard protocols to store and retrieve the offline information, keeping the P2PSIP scope restricted to storing resource mappings.

9. Pluggable DHT Algorithm Requirements

All dSIP peers MUST support the Chord pluggable DHT algorithm for compatibility. They MAY support additional pluggable algorithms. The requirements for new pluggable algorithms are defined in this section.

Pluggable algorithm MUST use Peer-IDs and Resource-IDs as defined in Hash Algorithms and Identifiers (Section 5.2) Pluggable algorithms are free to define what hash algorithms they support, but they MUST clearly specify what they are.

A resource with Resource-ID k will be stored by the peer with Peer-ID closest to the Resource-ID. The definition of closeness may vary in different DHT algorithms, but each DHT algorithm MUST guarantee Resource-ID searches converge to exactly one peer responsible for that portion of the namespace. As peers enter and leave, resources may be stored on different peers, so the information related to them is exchanged as peers enter and leave. Redundancy is used to protect against loss of information in the event of a peer failure.

Each new DHT algorithm MUST define a value for the dht-name parameter to be used in the dht-param parameter of the DHT-PeerID header, as defined in DHT Algorithms and the dht parameter (Section 5.4.3).

Each new DHT algorithm MUST define the valid BNF for the link-value used in the DHT-Link header, as defined in The DHT-Link header (Section 5.5).

10. Security Considerations

The goal of P2PSIP is to scale gracefully from ad hoc groups of a few people to an overlay of millions of peers across the globe. As such, there is no one security model that fits the needs of all envisioned environments; for the small network establishing a certificate chain is ludicrously difficult, while for a global network the unrestricted ability to insert resources and devise useful Peer IDs is a clear invitation to insecurity. Any P2PSIP protocol must offer a range of security models that can be selected according to the needs of the overlay.

10.1. Threat Model

Without other security, the attacker is able to generate an ID and become a valid peer in the system. They can see other peers and process certain queries. Attackers may wish to receive communications intended for other participants, prevent other users from receiving their messages, prevent large portions of the users from receiving messages, or send messages that appear to be from others. Users would like to be sure they are communicating with the same person they have previously talked to, to be able to verify identity via some out of band mechanism. Attackers may try to squat on all the good names. Users would like names that are meaningful to them. Attackers may have computers that are many times faster than the average user's. Attackers may be able to DOS other particular peers and make them fail. To make a robust DHT, many peers need to store information on behalf of the community. Peers may lie about this and not store the information. Attackers may wish to see who is communicating with whom and how much data is getting communicated.

Many of the threats to P2P SIP are also threats to conventional SIP. As such, P2P SIP imports much of its security from conventional SIP. However, because conventional SIP generally relies on secure servers to maintain the integrity of the system, modifications to those techniques are required to maintain the same level of security.

10.2. Protecting the ID Namespace

The fundamental protection that P2PSIP relies on is protecting the ID namespace. In particular, many of the attacks on P2PSIP require identifying a particular portion of the ID space and acquiring control of that space. This is a common vector both for attacks on a particular user, by obtaining control of the location in the overlay where the user is registered, and on the overlay itself, by means of a Sybil [19] attack when one is able to insert multiple identities at different locations on the ring.

The P2PSIP ID Namespace is considered protected when an attacker is not able to select an arbitrary Peer-ID and insert a peer at the location by convincing other peers to route traffic to them. This protects against hijacking and DoS attacks. The ID Namespace may also be protected by restricting admission to the overlay to some authorized (and trusted) set of individuals.

10.2.1. Protection Using ID Hashing

The default base security for P2PSIP determines Peer-IDs by hashing the peer's IP address and appending the port number. The security of this scheme depends on the ease with which an attacker can choose their own Peer-ID. Because the port number is only appended to the Peer-ID, an attacker gains nothing by selecting different ports on the same node. Assuming that the SHA1 hash used to calculate the Peer-ID is reliably random, the attacker's ability to succeed depends on the number of separate IP addresses that they are able to obtain from which to launch their attacks.

In the current predominantly IPV4 Internet, few attackers have access to more than a handful of IP addresses, perhaps a few hundred at worst. For a large-scale P2P system, this is unlikely to provide the ability to hijack a particular user ID or control a sufficient portion of the network to affect other peers, in particular when registrations are replicated at independent peers. Ultimately, however, a sufficiently skilled and provisioned attacker can compromise this scheme.

As the Internet migrates to IPV6, however, it is unclear that the assumption that few attackers have access to a significant range of IP addresses will remain true. Therefore, hashing IP addresses to

Peer-IDs is assumed to provide a diminishing amount of security in the future.

10.2.2. Cryptographic Protection

Stronger protection guarantees are possible by relying on cryptographic techniques to restrict the generation of peer IDs, either through requiring knowledge of a shared secret to calculate a valid hash or by issuing certificates through a central authority. These techniques are further described in Security for dSIP [9].

10.3. Protecting the resource namespace

The two primary vectors of attacks on resources in a P2PSIP overlay are inserting illegitimate resources into the overlay and corrupting the registrations for which a compromised peer is responsible.

For overlays that do not rely on certificates, once a peer has joined the overlay there are no restrictions on its ability to register resources. In an unsecured network, multiple peers can register the same resource (username) in the overlay. However, self-signed certificates [14] can be used to authenticate a user as the same user previously contacted with that certificate. Unless a conventional SIP authentication server is available, however, establishing identity upon initial contact is still a problem. One potential solution is for an overlay that is expected to persist over long time-frames to store the credentials of previous users for verification of a new registration. These techniques are beyond the scope of this document.

The second form of resource attack, which is really an ID attack, concerns the attacks that are possible when a peer has legitimately inserted itself into the overlay and is now responsible for storing resource registrations. Such an attack could occur through a corrupted peer or by an attacker who convinces the CA to issue them a certificate for a Peer-ID. In this case, the peer can corrupt any resource that is assigned to it. In the absence of certificates, the primary means of defense of such attacks is relying on the replication described in Section Section 5.2.2. By storing replicas of each registration on multiple peers and performing parallel searches for resource lookup, the searching peer protects itself from a single peer trying to corrupt the namespace.

Further protection from each attack vector is achieved by relying on certificates for resource authentication [9].

10.4. Protecting the Routing

The DHT forms a complex routing table. When a peer joins, it may contact a subversive peer that lies about the finger table information it provides. The subversive peer could do this to try to trick the joining peer to route all the traffic to a subversive group of peers. Prevention of this attack relies on protecting the namespace and (for hashed namespaces) identifying trusted bootstrap peers to use when joining.

Resource searches are protected from a single subversive peer through the use of parallel searches on replicated registrations. Similar protection could be achieved through performing parallel searches using multiple bootstrap peers for initial join, but such specification is beyond the scope of this draft. When possible, securing the namespace is a better solution.

10.5. Protecting the Signaling

The goal here is to stop an attacker from knowing who is signaling what to whom. An attacker being able to observe the activities of a specific individual is unlikely given the randomization of IDs and routing based on the present peers discussed above.

10.6. Protecting the Media

As with conventional SIP, all the media SHOULD be encrypted. Negotiating encryption for an end-to-end media session should be performed in the same manner for P2PSIP communications.

10.7. Replay Attacks

Defense against replay attacks is discussed in [9].

11. Open Issues

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

Still to be worked out are details of how P2PSIP names are disambiguated from conventional names that use DNS based routing.

12. Acknowledgments

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.

13. IANA Considerations

This document would require registering the following:

- o Option tag "DHT"
- o "DHT-Link" as a Header Field
- o "DHT-PeerID" as a Header Field
- o "peer" as a valid value for parameter user (?)
- o "Resource-ID" as a valid URI parameter (?)
- o "hmac-sha1" as an Identity-Info 'alg' parameter

[ToDo: This section needs to be revamped to include all the new BNF introduced]

14. Changes to this Version

While this is a -00 document, it has grown from the earlier drafts draft-bryan-sipping-p2p-xx. As such, we discuss the changes from the most recent version of that draft, -03.

1. The earlier draft has been split into a number of drafts:
    1. This draft, providing the background and overall concept, basic terminology for encoding P2P messages in SIP,
2. We have removed "-" from a number of headers and parameter names to shorten the overall length of the messages. Additionally, we have provided short versions for some strings in the syntax to help reduce message size.
3. We have attempted to use the new terminology defined in [2] wherever possible, and have attempted not to replicate definitions here. In particular, we have substituted the use of the term "peer" for "node"
4. As a consequence of the above, NodeID has been replaced with PeerID, both in text and in the actual defined messages sent over the wire.
5. We have made many changes to include details essential to using this in real deployed systems or clarifying difficult concepts; lessons learned from building a commercial application based on this draft.

6. Large parts of the description of how an initial overlay is formed were quite confusing as our description did not explicitly embrace the NULL predecessor concept of Chord. We have corrected this in the sections describing the algorithms.
7. A full and detailed example showing the startup of a 3 node system has been inserted into the examples section.
8. A new section has been added detailing early work on incorporating SIP identity into a P2P environment. This work is then used in the security section.
9. The security section has been thoroughly rewritten to reflect changes both in our thoughts and the thoughts of the P2PSIP working group as a whole.
10. We corrected a number of outright errors and typos pointed by a number of individuals, as mentioned in the acknowledgments.

15. References

15.1. Normative References

[1] 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.

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

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

[4] Rosenberg, J., "Simple Traversal Underneath Network Address Translators (NAT) (STUN)", draft-ietf-behave-rfc3489bis-05 (work in progress), October 2006.

[5] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-ietf-mmusic-ice-13 (work in progress), January 2007.

[6] Cooper, E., Matthews, P., Bryan, D., and B. Lowekamp, "NAT Traversal for dSIP", Internet Draft draft-matthews-p2psip-dsip-nat-traversal-00, February 2007.

[7] Zangrilli, M. and D. Bryan, "A Chord-based DHT for Resource Lookup in P2PSIP", Internet Draft draft-zangrilli-p2psip-dsip-dhtchord-00, February 2007.

[8] Zangrilli, M. and D. Bryan, "A Bamboo-based DHT for Resource Lookup in P2PSIP", Internet Draft draft-zangrilli-p2psip-dsip-dhtbamboo-00, February 2007.

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

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

[11] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 4474, August 2006.

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

[13] Rosenberg, J., Schulzrinne, H., and P. Kyzivat, "Caller Preferences for the Session Initiation Protocol (SIP)", RFC 3841, August 2004.

[14] Jennings, C., "Certificate Management Service for The Session Initiation Protocol (SIP)", draft-ietf-sip-certs-02 (work in progress), October 2006.

15.2. Informative References

[15] Bryan, D., Shim, E., and B. Lowekamp, "Use Cases for Peer-to-Peer Session Initiation Protocol (P2PSIP)", Internet Draft draft-bryan-sipping-p2p-usecases-00, November 2005.

[16] Bryan, D., Jennings, C., and B. Lowekamp, "SOSIMPLE: A Serverless, Standards-based, P2P SIP Communication System", Proceedings of the 2005 International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications (AAA-IDEA) '05, June 2005.

[17] Rosenberg, J., "Obtaining Relay Addresses from Simple Traversal Underneath NAT (STUN)", draft-ietf-behave-turn-02 (work in progress), October 2006.

[18] Cao, F., Bryan, D., and B. Lowekamp, "Providing Secure Services in Peer-to-Peer Communications Networks with Central Security Server", Internation Conference on Internet and Web Applications and Services (ICIW) '06, February 2006.

[19] Douceur, J., "The Sybil Attack", IPTPS '02, March 2002.

Authors' Addresses

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

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

Bruce B. Lowekamp
SIPeerior; William & Mary
3000 Easter Circle
Williamsburg, VA 23188
USA

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

Cullen Jennings
Cisco Systems
170 West Tasman Drive
MS: SJC-21/3
San Jose, CA 95134
USA

Phone: +1 408 421 9990
Email: fluffy@cisco.com

Acknowledgment

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