A Distributed Transport Function in P2PSIP using HIP for Multi-Hop Overlay Routing
draft-matthews-p2psip-hip-hop-00

Status of this Memo

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

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

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

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

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

This Internet-Draft will expire on December 18, 2007.

Copyright Notice

Copyright (C) The IETF Trust (2007).

Abstract

This document examines a P2PSIP architecture where the peer-to-peer (P2P) layer is separate from and lies below the SIP layer. We discuss the functions of the P2P layer in such an architecture, and focus in on the Distributed Transport function - the function that allows a peer to exchange messages with any other peer in the overlay, even in the presence of NATs. We list the features that the
Distributed Transport function needs to provide, and observe that the Host Identity Protocol (HIP) already provides a number of these features. We then propose extensions to HIP that allow it to provide the missing features. We discuss how a complete P2PSIP architecture can be built around HIP, and contrast this approach with other approaches for implementing a P2P layer. Two of the advantages of HIP approach are that (a) most existing applications can run in an overlay without needing any changes and (b) peer mobility and NAT traversal are handled in a way that is transparent to most applications.

Terminology

Descriptions of the basic concepts and terminology used in this document can be found in the P2PSIP Concepts and Terminology document [Concepts].
Table of Contents

1. Introduction .................................................. 4
   1.1. Distributed Database function .......................... 5
   1.2. Overlay Maintenance function ....................... 6
   1.3. Distributed Transport function ...................... 6
   1.4. Realizing the Distributed Transport function with HIP 8

2. Brief Introduction to HIP .................................. 9

3. Brief Introduction to our HIP extensions .................. 12

4. What are the alternatives? ................................. 13

5. Details of our Proposal .................................... 14
   5.1. Protocol Layering .................................... 15
   5.2. Peer IDs ............................................. 16
   5.3. Signaling ............................................. 16
   5.4. Sending Packets between Peers in the Overlay .... 17
      5.4.1. Routing Packets hop-by-hop through the Overlay 18
      5.4.2. Sending packets directly to the destination peer 18
   5.5. Security ............................................. 21

6. One Possible Implementation ............................... 22

7. IANA Considerations ....................................... 23

8. Security Considerations .................................. 23

9. Acknowledgments ........................................... 23

10. Informative References ................................... 23

Authors’ Addresses .......................................... 25
Intellectual Property and Copyright Statements ............. 26
1. Introduction

Consider the architecture for a P2PSIP peer shown in Figure 1.

<table>
<thead>
<tr>
<th>P2P Layer</th>
<th>SIP</th>
<th>Other applications ...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Distributed Database</td>
<td>Overlay Maintenance</td>
</tr>
<tr>
<td></td>
<td>Distributed Transport</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1

In this architecture, there is a P2P layer which is distinct from the SIP layer, and which provides services to the SIP layer and other applications. This P2P layer is internally divided into three parts, each of which provides a distinct function to the upper layers. These three functions are:

- Distributed Database. This allows peers to store and retrieve information. The initial envisioned use of this database is to store Address-of-Record (AoS) to Contact mappings for users, but it seems likely that this database will be used to store other things as P2PSIP evolves.

- Overlay Maintenance. This establishes and maintains peer connections, distributes overlay configuration information to peers, and does anything else required to maintain the overlay.

- Distributed Transport. This allows a peer to send an arbitrary packet to any other peer in the overlay, even if the destination peer is behind one or more NATs. This is the most basic function of the three, and is used by the other two functions.

The SIP layer utilizes the functions provided by the P2P layer to set up multimedia sessions between peers. SIP queries the Distributed Database function to resolve an AoR to one or more Contact addresses, and then uses the Distributed Transport function to deliver SIP messages to the remote peer(s). Note that SIP and other applications can access the Distributed Transport function directly without going through the other two functions.

It is important to note that we are proposing that the P2P layer provide these functions to all upper-layer protocols, not just SIP.
The authors strongly believe that people will want to run protocols other than SIP over P2PSIP overlays, and providing a solution that works only for SIP will just encourage people to run these other protocols over SIP - a solution which goes directly against [RFC4485].

This architecture proposal is not new. The initial suggestion to use this architecture for P2PSIP was made by us two years ago in [IPCom] and [Industrial], and has been explored by others in some detail in [P2P-Arch] and [P2PCommon]. The contribution of this document is not in suggesting the architecture, but in making a concrete suggestion for how to realize it that leverages a large body of existing work.

This architecture stands in contrast to the dSIP architecture [dSIP], where there is not a distinct P2P layer, but instead the SIP and P2P layers are merged and the functions of the P2P layer are implemented using an extended version of SIP.

In the following subsections, we examine in more detail the functions that the P2P layer provides in this architecture.

1.1. Distributed Database function

This function provides a way for upper-layer applications to provide and retrieve data that is actually stored by distributing the data out to the peers in the overlay.

In particular, the Distributed Database function provides the following:

- Distribution of data across peers in the overlay in a way such that no one peer needs to store all the data (unless there is only one peer in the overlay);
- Replication and shuffling so that data is not lost if one or more peers leave the overlay;
- Security (to the extent possible) to verify the origin of the data and guard against malicious data modification by other peers;
- Put and Get operations to store and retrieve the data from the distributed database.

There have already been some proposals for how the Distributed Database function might be realized. For example, [P2PCommon] proposes Insert, Lookup, and Remove messages that implement these many of the above features. We believe that these messages could be
easily modified to work with the Distributed Transport design described here.

1.2. Overlay Maintenance function

The Overlay Maintenance function provides the controls that causes the peers in the overlay to function together in a harmonious way.

For example, the Overlay Maintenance function provides the following:

- Admission of peers to the overlay, including checking the credentials of peers to make sure they are authorized to join;
- Controlling the creation of connections in the overlay to ensure that the appropriate pattern of connections exists for efficient routing and lookup;
- Distributing information about the overlay that needs to be known by all (or a subset) of the peers. This might include the name of the overlay, the values to use for adjustable parameters, encryption keys for data that all peers can read but nodes outside the overlay cannot, etc. This information is likely given to a peer when it joins the overlay, but there may be ways an administrator can change certain values without having to break up the overlay and allowing it to re-form.

This document does not propose an Overlay Maintenance protocol, leaving this to future work. However, later in this document we describe the role of the Overlay Maintenance protocol in driving the routing feature of the Distributed Transport function.

1.3. Distributed Transport function

The Distributed Transport function provides a way to uniquely identify peers and to deliver messages to an arbitrary peer in the presence of NATs and mobile peers.

The presence of NATs has a major influence on this function, since NATs often hinder two peers from exchanging data directly. The proposed approach for solving this problem is to establish a partial mesh of connections between peers, and then allow data to be sent indirectly between peers by sending it along existing connections in the overlay. To make this possible, there must be a way to identify a peer (a peer ID), a way to establish and maintain connections, and a way to add the destination peer ID to the packet. In essence, the overlay forms a network, with peer IDs serving as addresses, connections serving as links, peers serving as routers, and the tag serving as a network layer header.
Having peer IDs also makes it possible to gracefully handle mobile peers. If a peer changes its IP address, then this could be considered equivalent to the peer leaving the overlay and later rejoining with a new IP address, but it is better if this could be viewed as simply a change in the address used to contact the peer.

Providing these functions at the P2P layer means that applications themselves do not need to worry about NAT traversal and mobility. This is a big advantage over competing approaches that require each application to handle these on their own.

The approach mentioned above provides datagram delivery, but to be useful, the Distributed Transport function must also provide all the usual transport layer services that applications depend upon. For example, the Distributed Transport function must provide services like TCP and TLS. If these services are not provided, then the P2PSIP WG will have to redo a large collection of SIP-related standards that depend on these services being available.

Thus the Distributed Transport function provides the following:

- **Peer IDs**: A unique identifier for each peer in the overlay.

- **Network layer**: The ability to deliver a message to an arbitrary peer in the overlay. In our view, this involves adding a header to the message that specifies the destination peer ID and then routing that message along existing connections in the overlay to the destination peer.

- **Signaling**: The ability to add, maintain, and remove connections in the overlay. The signaling procedures must work in the presence of NATs.

- **Bootstrapping**: The ability for a peer that is not currently a member of the overlay to locate and establish an initial connection to a peer in the overlay.

- **Transport layer**: The usual transport layer functions such as port numbers, reliable in-order delivery of messages (if desired), and the segmentation of user data into Path-MTU-sized chunks (if desired). This also includes transport layer security (TLS and DTLS) if desired.

- **Mobility and Multihoming**: The ability for a peer have multiple IP addresses and to change these addresses dynamically while remaining a member of the overlay. In make-before-break scenarios (= adding new addresses before losing all the old addresses), this is seamless; in break-before-make scenarios, connections go down
and must be re-established but the peer remains part of the overlay.

| o Security: Message integrity and sequencing to prevent outsiders and intermediate peers from corrupting or replaying messages; encryption to prevent the message body from being read by outsiders or intermediate peers; protection against DoS attacks from outsiders or (to the extent possible) from intermediate peers.

The following figure shows a simple example of some of these concepts.

```
   O
 Peer E    / | \
    |   O Peer D / | \
    | O Peer C / | \\
    \
 Peer B O Peer H O Peer G
   /
 Peer A
```

Figure 2

Figure 2 shows a number of peers arranged in an overlay network. Each peer in the network is behind its own NAT. Each peer has one or more connections to other peers in the overlay. If peer H wants to send a packet to peer B, it could try to send the packet directly, but most likely the filtering property of B’s NAT would prevent the packet from getting through. So peer H has two options: (a) it can send the packet to peer A which then forwards it to peer B, or (b) it can set up a direct connection to peer B, using ICE-like signaling procedures [ICE], and then send the packet directly to B.

1.4. Realizing the Distributed Transport function with HIP

In this document, we propose to realize the Distributed Transport function with an extended version of the Host Identity Protocol (HIP) [HIP-Base] currently being developed in the HIP WG. We describe how HIP currently provides a number of the Distributed Transport features listed in the previous section, and then describe how to extend HIP to provide the remaining features. We contrast this approach of using HIP with the approach of producing a new protocol from scratch,
and conclude that HIP is such a good fit that any compelling new protocol would end up stealing many ideas from HIP.

The current version of this document is not a fully fleshed-out proposal, but rather a high-level presentation of the big picture. In many cases, we only describe the key ideas behind a proposed HIP extension, or the key ideas on how a Distributed Transport feature can be realized using either existing or proposed HIP features. We have taken this approach in part to keep the document short and readable, but mostly because in many cases we have not worked out the details. In addition, some of this work is perhaps best done in the HIP WG rather than the P2PSIP WG. We expect that future revisions of this document and/or follow-on documents will provide more details.

2. Brief Introduction to HIP

In this section, we give a brief introduction to HIP and how it is used in our proposal. This section is especially targeted at those who know little or nothing about HIP. The goal is to give the reader a sense that HIP has a lot to offer P2PSIP.

The Host Identity Protocol (HIP) is an alternative to the dual use of IP addresses as "locators" (routing labels) and "identifiers" (host identifiers). In HIP, the transport layer is decoupled from the network layer by introducing an identifier for a host which is independent of the host’s IP address(es). Though this decoupling, the transport layer and the applications above it are mostly insulated from changes in IP addresses. This host identifier concept of HIP is very similar to the peer ID concept of P2PSIP.

In HIP, hosts are identified using two closely-related concepts:

- A Host Identity (HI), which is the public half of a public/private key pair; and
- A Host Identity Tag (HIT), which is a 128-bit SHA-1 hash of a Host Identity.

An HI is the definitive identification for a host. HIs are long-lived, but it is easy for a host to have multiple HIs, and it is possible for hosts to create HIs without needing to access a central server.

The HIT is a compact (128-bit) shorthand for the HI with the following properties:
The HIP protocol itself is a signaling protocol for setting up, maintaining, and tearing down security associations between two hosts. Associations are created using a four-packet exchange. The first party is called the Initiator and the second party the Responder. The four-packet design helps to make HIP DoS resilient. The protocol exchanges Diffie-Hellman keys in the 2nd and 3rd packets, and authenticates the parties in the 3rd and 4th packets. Once the association is established, HIP defines other procedures for maintaining this association, even in the case where one or both ends change their IP address.

To allow the HIP association to traverse intervening NATs, HIP uses a variation of the ICE protocol [ICE]; see [NAT-Traversal-for-HIP].

A HIP association is logically a connection between two hosts. Once an association between two hosts is set up, HIP multiplexes all application-level protocols over the association. This is done by running the standard Internet transport protocols over the association, and using port numbers for demultiplexing in the usual manner.

In our proposal, HIP is used in two different ways: (a) the HIP signaling procedures are used as an important first step in setting up and maintaining a connection in the overlay, and (b) HIP is extended to act as an encapsulation protocol for carrying upper-layer application data hop-by-hop through the overlay.

The HIP header is illustrated in Figure 3.
The header contains the following fields:

- **Next Header**: Specifies the protocol that follow (lies above) HIP.
- **Header Length**: The total length of the header, including any optional HIP parameters.
- **Packet Type**: There are currently 8 different HIP packet types defined.
- **Version**.
- **Checksum**: A checksum over the header.
- **Controls**: A set of one-bit flags. Only one is currently defined.
- **Source (sender) and Destination (receiver) HITs**.
- **HIP Parameters**: Optional TLVs that carry additional information. A number of TLVs are currently defined.
3. Brief Introduction to our HIP extensions

The previous section described the features that HIP offers today and how it provides some of the features of the Distributed Transport function. In this section, we sketch the extensions to HIP required to provide the remaining features. The goal of this section is to give a quick overview of these extensions; more details are provided later.

The HIP extensions that we propose in this document are:

- Encapsulation of higher-level messages. HIP as currently defined is a signaling protocol only. To extend HIP to serve an encapsulation protocol for higher-layer messages to transport them hop-by-hop through the overlay, we exploit the fact that HIP header (Figure 3) already has a Next Protocol field. Thus encapsulating higher-layer messages is simply a matter of defining a codepoint for a new Packet Type (which we call the Data packet) which is used to carry the higher-layer messages.

- Hop-by-hop routing through an overlay. HIP as currently defined is a protocol for setting up a point-to-point security association between two hosts. To extend it to provide multi-hop routing between peers in an overlay, we exploit the fact that the HIP header (Figure 3) contains the source and destination HITs (= peer IDs). Thus doing multi-hop routing is simply a matter of defining how to forward a HIP message in the case when the peer receiving the message is not the destination peer listed in the header.

- Bootstrapping. In [Bootstrap], the authors described procedures that allowed a joining peer to locate and establish a connection to an admitting peer in an overlay. Those procedures were defined using SIP as the signaling protocol, but these procedures can also be realized using HIP as the signaling protocol.

- Efficient transport of SRTP. HIP inserts an extra layer into the standard networking stack, and two layers when there is a NAT between the two peers. For protocols like the Secure Real-Time Protocol (SRTP), these extra layers can cause problems. We show how the usual protocol stack (e.g., SRTP / UDP / IP) can be used in these situations, while maintaining the NAT traversal, multi-homing, and mobility properties of our version of HIP.

More details on these extensions and other aspects of our proposal can be found in Section 5.
4. What are the alternatives?

Before we jump into the details of our proposal, it is worth considering what an alternative design for the Distributed Transport layer would look like if the P2PSIP group was not to use HIP but design one from scratch. For the authors, it was a real breakthrough when we realized that any protocol we designed from scratch to be a Distributed Transport layer would likely re-invent much of HIP.

To start with, consider the format of a peer ID. In [dSIP] and elsewhere, it is proposed that P2PSIP use 160-bit peer IDs. To use these peer IDs, [dSIP] and [Bootstrap] propose to extend the URI scheme of SIP to express peer IDs, perhaps by adding a "peerid=" parameter to the URI. There are two problems with this approach: (a) every application that wants to work in an overlay has to be extended to understand the new URI scheme, and (b) new procedures have to be defined to describe how an application resolves this URI. A counter-argument might be made that many DHTs today are defined to work with 160-bit hashes, but the authors believe that all the major DHTs today can be easily adapted to work with a 128-bit peer ID.

By contrast, the approach of HIP is to make a peer ID (= HIT) look like an IPv6 address. With this approach, in most cases the existing approaches for resolving a URI to an address continue to work if (behind the scenes) a peer ID is returned instead of an address. As we show below, this means that most applications need no changes to work in an overlay. From this, we conclude that the advantages of making a peer ID look like an IPv6 address are substantial, and any alternate proposal for P2PSIP would need strong reasons to take a different approach.

Next, consider the design of the "network layer" header, and consider the fields that would be needed in this header if the P2PSIP WG was to design its own protocol header. Most likely, these fields would include:

- Source and destination peer ID (required for routing around the overlay);

- Demux field for indicating the higher-layer protocol (required to determine the upper-layer protocol);

- Protocol version number;

- Packet type field (if the protocol also does signaling or other things);
o Optional TLVs for extensibility;

o Some way to detect the end of the header when optional TLVs are present; for example, a header length field.

Comparing with the HIP header in section, we see that only the Header Checksum and Controls fields might be eliminated, and even these fields can be easily argued for. From this, we conclude that the HIP header extremely well-suited for the Distributed Transport layer.

Next, consider the signaling protocol. The basic functions of the HIP signaling protocol (setting up, maintaining, and tearing down connections, handling endpoint mobility, reporting errors, etc) are the same as a dedicated P2PSIP protocol would need. Though it is possible, perhaps even likely, that a P2PSIP design team would make some different design choices, the resulting protocol would likely have all the same basic properties.

Finally, consider the transport layer functions. In HIP, these are performed by the existing transport layer protocols (TCP, UDP, TLS, etc) using the existing APIs (sockets, etc.), exploiting the fact that HITs look like IPv6 addresses. In this way, little or no changes are required to existing applications. This makes for a very compelling story in comparison with the alternatives of developing new APIs and/or new protocols.

5. Details of our Proposal

This section gives the details of our proposal for using an extended version of HIP for the Distributed Transport function of P2PSIP.

While reading this proposal, there are a few facts that reader should keep in mind:

o The proposal does NOT require the underlying network to be IPv6. Though peer IDs look like IPv6 addresses at the application layer, the underlying network addresses can be IPv4, IPv6, or a mixture of the two.

o Only peers and bootstrap servers need to run the HIP-related protocols. No changes are required on other nodes in the network (e.g., routers, client-server SIP nodes, or other nodes that interact with the overlay).

The following sections give a high-level view of the proposal. More details will be provided in subsequent versions and/or separate documents.
5.1. Protocol Layering

Figure 4 shows the fundamental protocol layering in our proposal.

<table>
<thead>
<tr>
<th>IP4</th>
<th>IP6</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4</td>
<td>IPv6</td>
</tr>
<tr>
<td>UDPv4</td>
<td>UDPv6</td>
</tr>
<tr>
<td>HIP or ESP</td>
<td></td>
</tr>
<tr>
<td>TCPv6, UDPv6, TLS, etc.</td>
<td></td>
</tr>
<tr>
<td>Distributed Database : Overlay Maintenance</td>
<td></td>
</tr>
<tr>
<td>SIP</td>
<td>Other applications ...</td>
</tr>
</tbody>
</table>

Figure 4

In Figure 4, the Distributed Transport box of Figure 1 has been replaced by three sub-layers. The upper sub-layer is the existing Internet transport layer, consisting of protocols such as TCP, UDP, SCTP, DCP, etc along with extensions such as TLS and DTLS. These are the v6 versions of these protocols, since HITs (peer IDs) look like IPv6 addresses.

The middle sub-layer is the HIP/ESP layer. HIP is used for signaling and for encapsulation of data packets in multi-hop scenarios, while ESP (Encapsulated Security Payload) [HIP-ESP] is used for encapsulation in single-hop scenarios -- we discuss this in more detail below.

The lower sub-layer is a UDP encapsulation layer. This layer is present because most NATs, firewalls, and other middleware boxes today do not understand HIP and will usually drop a packet if the protocol above the IP layer is not TCP or UDP. Placing a UDP header between IP and HIP will allow HIP packets to traverse these boxes. This layer is used only when required. Using ICE-like connectivity checks, HIP detects if packets without this encapsulation layer can make it through and eliminates this layer when it is not needed.

This stack runs over either IPv4 or IPv6. A peer can have both IPv4 and IPv6 interfaces, and connections in the overlay can be a mixture of these two protocols.
NOTE: Readers concerned about how to implement Figure 4 may wish to jump ahead to Section 6 before reading further.

5.2. Peer IDs

Host Identities could be assigned to peers in at least two different ways. One way is for peers to generate their own public/private key pairs. Another way is to allocate them to peers, perhaps in conjunction with a set of credentials, using a centralized allocation system. The pros and cons of these and other schemes requires further investigation.

Once a Host Identity is allocated to a peer, the peer uses the standardized method to form its HIT [HIP-Base].

A HIT is the typical way to identify a peer in the overlay. Because a HIT fits in an IPv6 address, in many cases applications need not be aware that they are talking to a peer in an overlay, and many IPv6-ready applications can run in an overlay without changes. Consider an application that uses the IPv6 form of the socket API, uses HITs to identify peers on the overlay, and uses IPv4 addresses (in IPv4-in-IPv6 format) and/or IPv6 addresses to identify nodes off the overlay. In many situations, the application can freely mix these three formats internally, leaving the transport and HIP layers to sort out the differences. The exceptions are cases where the application would otherwise do something like send a HIT to an IPv4-only node not on the overlay.

5.3. Signaling

In our proposal, there are two layers of signaling involved in establishing, maintaining, and terminating connections in the overlay. The HIP layer is responsible for establishing, maintaining, and terminating HIP associations with other nodes. The nodes may be peers in the overlay, or they may be ordinary nodes with which a HIP association is desired. The Overlay Maintenance layer is responsible for admitting some of these HIP associations to the overlay, and for ensuring that the pattern of connections in the overlay follow the pattern required for the DHT or other protocol. In this section, we discuss HIP signaling for overlays in more detail, and leave the discussion of Overlay Maintenance signaling to other documents.

Establishing a new HIP association within an overlay falls into one of two cases: (a) the initiating peer is not currently in the overlay and is trying to establish its first connection to another peer in the overlay, and (b) the initiating peer is already in the overlay. The basic format of the signaling exchange is the same in both cases; the difference is in how the HIP signaling messages are routed.
between the two peers.

In case (a), procedures similar to those in [Bootstrap] are used. [Bootstrap] defines two mechanisms for a joining peer to locate an admitting peer: using a Bootstrap Server and using multicast. HIP already a mechanism similar to the Bootstrap Server mechanism (the RVS mechanism) which is used to locate a single node -- in our proposal, this mechanism is extended to work with overlays. The key idea is to identify the overlay either by name, or by assigning a HIT to the overlay itself. In that way, the bootstrap peers can register with the Bootstrap Server using the overlay name or HIT, and the Bootstrap Server can route HIP I1 packets (= the first packet in the HIP signaling exchange) received from the joining peer to a bootstrap peer associated with the overlay. For the multicast mechanism, a similar approach is used: a multicast I1 packet specifying the overlay to join is sent out by the joining peer, one or more bootstrap peers reply, and the joining peer selects one to continue the exchange with.

In case (b), the signaling messages are delivered to the remote peer by routing them hop-by-hop through the overlay (section 4.4.1). The initiating peer places the HIT of the remote peer into the I1 message and sends the I1 message to its direct neighbor which is closest to the remote peer, and the I1 message is then routed hop-by-hop to the remote peer. In this way, the originator does not need have a priori knowledge of the remote peer’s IP address, and the signaling messages can be delivered even if the remote peer is behind a NAT or firewall.

At any time, a given peer may have some associations which are a part of one overlay, some associations which are part of other overlays, and some associations which are not part of any overlay (or equivalently, a part of a 2-node overlay only). The question of whether a given HIP association can be simultaneously part of two different overlays is for further study.

5.4. Sending Packets between Peers in the Overlay

There are two ways to send a packet to another peer in the overlay: send it on a direct connection to the remote peer, or send it hop-by-hop through the overlay. A peer typically uses hop-by-hop routing when it has only a small amount of data to transfer to the remote peer (for example, a Distributed Database update or a SIP INVITE transaction), and sets up a direct connection when it has a larger amount of data to transfer (for example, an RTP session).
5.4.1. Routing Packets hop-by-hop through the Overlay

To route a packet hop-by-hop through the overlay, it must have a HIP header. In this HIP header, the sender field gives the HIT of the peer sending the packet, and the receiver field gives the HIT of the peer to which the packet is destined -- this might be a peer that is multiple hops away.

The HIP packet might be one of the existing packet types used to set up and maintain HIP associations, or it might be a new packet type called Data that is used to encapsulate messages from higher-layer protocols and carry them hop-by-hop through the overlay. This new packet type has the HIP header shown in Figure 3, a packet type of "Data" (codepoint is TBD), and the Next Protocol field in the header is used to indicate the encapsulated protocol.

We then extend HIP with the concept of multi-hop routing. When a HIP packet arrives at a peer, the packet is delivered to the HIP layer which checks if the destination HIT is a HIT that belongs to this peer. If not, then the peer tries to forward the packet. To do this, the peer must decide which of its (directly connected) neighboring peers to forward the packet to. This is done by having the HIP layer consult a table, called the HFIB (HIP Forwarding Information Base), which plays a role similar to the FIB table used in IP forwarding by routers. The calculation of the HFIB is done by the Overlay Maintenance layer and downloaded to the HIP layer.

The Overlay Maintenance layer constructs the HFIB using the principle of greedy routing, where at each hop, packets are forwarded to the neighboring peer whose peer ID is the closest match to the destination peer ID. This is the routing approach used in most DHT algorithms (Chord, Bamboo, Kademlia, etc). The Overlay Maintenance layer makes this routing algorithm efficient by adding the appropriate connections to the overlay. More discussion of this approach can be found in [NATs-and-Overlays].

It is possible for a given peer to be a member of multiple overlays. It is also possible for a peer to have HIP associations with nodes that are not part of an overlay. In these cases, a peer needs to know on which overlay (or otherwise) a given packet should be forwarded. One way to solve this problem is to include the overlay ID in a TLV in the HIP header. This is an area of ongoing investigation.

5.4.2. Sending packets directly to the destination peer

The second way to send a packet to a peer in the overlay is to establish a direct connection to the remote peer, and then send the packets directly.
When sending a packet on a direct connection to the destination peer, the relatively large HIP header (40 bytes) can be replaced with something smaller. In this document, we discuss two replacements: the first is currently defined for use with HIP, while the second is a proposed extension.

The first alternative is shown in Figure 5.

```
<table>
<thead>
<tr>
<th>SIP</th>
<th>Other applications ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>.............</td>
<td>.............</td>
</tr>
<tr>
<td>Distributed Database : : Overlay Maintenance</td>
<td></td>
</tr>
<tr>
<td>............. : : .............</td>
<td></td>
</tr>
<tr>
<td>............. : : .............</td>
<td></td>
</tr>
<tr>
<td>TCPv6, UDPv6, TLS, etc.</td>
<td></td>
</tr>
<tr>
<td>............. : : .............</td>
<td></td>
</tr>
<tr>
<td>............. : : .............</td>
<td></td>
</tr>
<tr>
<td>ESP</td>
<td>UDPv4</td>
</tr>
<tr>
<td>............. : : .............</td>
<td></td>
</tr>
<tr>
<td>UDPv4        : : UDPv6</td>
<td></td>
</tr>
<tr>
<td>IPv4</td>
<td>IPv6</td>
</tr>
</tbody>
</table>
```

Figure 5

Here HIP has been replaced with ESP (Encapsulated Security Payload) [HIP-ESP]. ESP serves two functions when used in this way:

- It provides optional encryption, optional message integrity, and protection against replay attacks. The pros and cons of using ESP vs. TLS/DTLS for this purpose in P2PSIP overlays in an area of ongoing investigation.

- It provides a field, called the SPI (Security Parameter Index), which is used as a shorthand for the (source, destination) HIT pair. In this way, the receiver can determine the source and destination HITs to associate with the packet.

Following the lead of the HIP WG, this protocol stack is the default when sending a packet directly from the source to the destination.

The advantage of the protocol stack in Figure 5 is that it provides a smaller header (8 bytes for ESP vs. 40 bytes for HIP), while maintaining the separation between the transport layer and the IP layer that allows the IP addresses to change without affecting the transport layer. For most applications, this protocol stack
represents a good tradeoff between efficiency and flexibility.

However, for some applications, the protocol stack in Figure 5 is inappropriae. A good example is SRTP, where a small header is very important, where there has been a fair amount of work on compressing the header further [RFC3095], and where the security properties of ESP are unnecessary since these properties are already provided by the application protocol.

For those applications, a second protocol stack is available, as shown in Figure 6.

```
<table>
<thead>
<tr>
<th>SRTP</th>
<th>Other apps...</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2P Layer</td>
<td>UDP(v4/v6)</td>
</tr>
<tr>
<td></td>
<td>IP(v4/v6)</td>
</tr>
</tbody>
</table>
```

Figure 6

Here the default protocol layering on direct connections (shown in Figure 5) has been replaced with an alternative layering. This is not a general-purpose layering; this layering must be explicitly negotiated by the two peers before it can be used and is available only for the specific combinations of (local HIT, local port, remote HIT, remote port, protocol=UDP) that have been negotiated. For all other combinations, the two peers continue to use the layering of Figure 5.

When a packet is received at a peer with this layering, the combination of (local IP address, local port, remote IP address, remote port, protocol=UDP) is used to map this packet to a specific (local HIT, local port, remote HIT, remote port, protocol=UDP) combination. In this way, both the HIT pair and the destination application are identified.

To negotiate this usage between two peers, one end (peer X) sends a HIP message to the other (peer Y) saying that it would like to negotiate an alternative protocol layering for a particular UDP port combination. Peer X includes a set of ICE candidates to use for the alternative layering; in ICE terms this can be viewed as another media stream between the two peers (where the HIP association is the primary media stream). If peer Y is agreeable, it replies with its own set of candidates, and the two peers then run connectivity checks.
to select a valid pair. Later, if one endpoint changes its IP address, the two peers negotiate a new valid candidate pair.

Since the use of this alternative protocol layering requires extra HIP messaging between the two peers to establish and maintain the additional "media stream", its use is recommended only in situations where the alternate protocol layering is important. In most situations, the default protocol layering of Figure 5 is quite sufficient.

From a HIP protocol perspective, this mechanism can be viewed as an instance of a more general mechanism for negotiating alternative protocol layerings. However, it is worthwhile noting that the details of doing a similar layering for TCP are significantly more complex. Consider the case where peer Y changes IP address when peer X has a number of unacknowledged segments outstanding. The sequence numbers of the new TCP connections must be related back to the old TCP connection to allow segments on the new connection to acknowledge segments on the old connection. The details and even the desirability of supporting this is left for future study.

5.5. Security

Security in HIP can be divided into two areas. The first is the security of the HIP protocol itself, the second is the security provided to the upper layers.

For the first, HIP currently provides encryption, message integrity, and protection against replay and denial-of-service attacks against HIP itself. We believe that these mechanisms extend in a straightforward way to multi-hop message exchanges, though we have not yet investigated all the details.

For the second, more investigation is needed to determine whether the security of application protocols should be provided by the HIP/ESP layer, provided at the transport layer by mechanisms such as TLS/DTLS, or provided at the application layer. The answer will probably be application-dependent. For SRTP, protection at the application layer seems appropriate. For SIP, protection at the transport layer seems appropriate, since SIP is already defined to use TLS over TCP.

For many applications, there is an interesting question of whether TLS or ESP is most appropriate. ESP seems to provide security only on an overlay-hop-by-overlay-hop basis, while TLS provides end-to-end security even across multiple overlay hops. ESP may be appropriate if the goal is to protect against outside attacks, while TLS may be more appropriate if the goal is to also protect against attacks from rogue peers.
6. One Possible Implementation

Consider implementing this proposal on a device which is IPv4-only and has a networking stack built into the OS that you cannot change. One way to do this is shown in Figure 7.

```
| (S)RTP | SIP | Other apps |
|________|_________|

| Distrib DB | Overlay Maint |
|____________|_______________|

Socket API (v6) .................................

| TCPv6 | UDPv6 |
|_______|_______|
| HIP / ESP | |

User Space

Socket API (v4) .................................

| TCPv4 | UDPv4 |
|_______|_______|
| IPv4 |

Kernel Space

Figure 7

In Figure 7, a standard IPv4 stack is built into the kernel and is accessed via the IPv4 version of the socket API. The HIP/ESP layer, with a second copy of the TCP/UDP layer, is located in user space and is accessible via the IPv6 version of the socket API. The HIP/ESP layer uses the socket-v4 interface into the kernel to send and receive packets. (Note: the v6 and v4 versions of the TCP and UDP protocols differ only in how their checksums are computed). The Distributed DB and Overlay Maintenance protocols live above the socket-v6 interface and uses that API to send and receive packets. Finally, the P2PSIP applications (SIP, (S)RTP, etc.) use the services of all the lower layers. If there is just one process that participates in the P2PSIP overlay, then all the layers shown in user space could be bundled together in that process.

Open-source code for many of the pieces in this diagram are available today (albeit without the HIP extensions described above).
7. IANA Considerations

The present version of this document introduces no new IANA considerations.

8. Security Considerations

The present version of this document gives only a high-level description of the proposal. A detailed security analysis will be provided in subsequent versions and/or related documents that describe the detailed mechanisms.

9. Acknowledgments

The authors thank Spencer Dawkins, Dean Willis, Kevin Chen, and Scott Hutchens for their helpful comments on this document.

10. Informative References

[Bootstrap]

[Concepts]

[HIP-Base]

[HIP-ESP]

[ICE]

[IPCom]
Johnston, A., "SIP, P2P, and Internet Communications", Internet Draft draft-johnston-sipping-p2p-ipcom-00,
January 2005.

[Industrial]

[NAT-Traversal-for-HIP]

[NATs-and-Overlays]

[P2P-Arch]

[P2PCommon]

[RFC3095]

[RFC4485]

[dSIP]
Authors’ Addresses

Eric Cooper
Avaya
1135 Innovation Drive
Ottawa, Ontario  K2K 3G7
Canada
Phone: +1 613 592 4343 x228
Email: ecooper@avaya.com

Alan Johnston
Avaya
St. Louis, MO  63124
USA
Email: alan@sipstation.com

Philip Matthews
Avaya
100 Innovation Drive
Ottawa, Ontario  K2K 3G7
Canada
Phone: +1 613 592 4343 x224
Email: philip_matthews@magma.ca
Full Copyright Statement

Copyright (C) The IETF Trust (2007).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgment

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