Service Extensible P2P Peer Protocol
draft-jiang-p2psip-sep-00.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any
copyrighted 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

This Internet-Draft will expire on March 4, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2007).

Abstract

This document proposes a Service Extensible Protocol (SEP), which is
spoken between P2PSIP peers. SEP uses a flexible forwarding
mechanism to avoid congestion in the overlay. It also proposes a
general service discovery method and a built-in NAT traversal
mechanism. By using these methods, SEP tries to improve the success
rate and reduce the latency of the transaction.
Table of Contents

1. Introduction ................................................. 4
2. Overview ................................................. 4
   2.1. Routing States ........................................ 5
   2.2. Data Operations ....................................... 6
   2.3. Data Transfer During the Overlay Churn .................. 6
3. Terminology .............................................. 7
4. Design Choice ............................................. 7
   4.1. Routing Mode .......................................... 8
   4.2. Service Discovery ..................................... 9
   4.3. Processing Status Advertisement from the Downstream Peers ............................................. 9
   4.4. An flexible Forwarding Mechanism ......................... 10
   4.5. End-to-end Reliability ................................ 10
5. Message Syntax ........................................... 11
   5.1. Message Header ...................................... 11
   5.2. Message Attribute ................................... 13
      5.2.1. TLV Format ....................................... 13
      5.2.2. Peer Address Info ................................ 13
      5.2.3. Peer service capability ............................ 14
      5.2.4. Peer-ID ........................................... 15
      5.2.5. Source Peer Info .................................. 15
      5.2.6. Destination Peer Info .............................. 16
      5.2.7. Resource Info ..................................... 16
      5.2.8. Negotiation Info .................................. 18
      5.2.9. Response Attribute ................................ 18
      5.2.10. Service Peer Info ................................ 18
      5.2.11. Relaying Peer Info ................................ 20
      5.2.12. Source Reflexive Address attribute ............... 20
      5.2.13. Routing table ..................................... 21
      5.2.14. Status Info ...................................... 21
      5.2.15. Event Info ....................................... 21
      5.2.16. Update Type ...................................... 22
6. Peer Protocol Message ..................................... 22
   6.1. Overlay Maintenance ................................... 22
      6.1.1. JOIN ............................................ 22
      6.1.2. LEAVE ........................................... 23
      6.1.3. KEEPALIVE ....................................... 24
      6.1.4. NOTIFY ........................................... 25
      6.1.5. UPDATE ........................................... 26
      6.1.6. SearchPeer ....................................... 26
      6.1.7. TRANSFER ......................................... 27
6.2. Data Operations ......................................... 28
      6.2.1. PUT ............................................. 28
      6.2.2. GET ............................................. 28
      6.2.3. REMOVE .......................................... 29
6.3. Additional messages ...................................... 29
6.3.1. LookUpServicePeer ........................................ 29
7. Forwarding Operations .......................................... 31
  7.1. Request Forwarding .......................................... 31
  7.2. Response Forwarding .......................................... 32
    7.2.1. Response Forwarding on the Destination Peer ............ 32
    7.2.2. Response Forwarding on the Intermediate Peer ......... 33
8. General Peer Behavior ............................................ 33
  8.1. Source Peer Behavior ......................................... 33
    8.1.1. Generating the Request and Sending the Request ....... 33
    8.1.2. Processing the Response ................................ 34
  8.2. Intermediate Peer Behavior .................................. 35
    8.2.1. Request Processing ...................................... 35
    8.2.2. Response Processing ..................................... 36
  8.3. Destination Peer Behavior .................................... 36
    8.3.1. Request Processing ...................................... 36
    8.3.2. Response Generation and Sending the Response ......... 36
9. NAT Traversal .................................................... 36
  9.1. Gather ICE candidates ....................................... 37
  9.2. Response from the Destination Peer to the Source Peer ... 38
    9.2.1. How to Make JOIN Response Return to Source Peer ..... 38
  9.3. Exchange Candidates for the Direct Communication .......... 39
10. Service Discovery ................................................ 39
11. Security Considerations ......................................... 40
12. IANA Considerations ............................................. 40
13. Acknowledgements ............................................... 40
14. References ...................................................... 40
    14.1. Normative References .................................... 40
    14.2. Informative References ................................... 41
Authors’ Addresses .................................................. 41
Intellectual Property and Copyright Statements .................... 42
1. Introduction

This document proposes a Service Extensible Protocol (SEP), which is spoken between P2PSIP peers. The SEP conforms to the definition of P2PSIP Peer protocol in [concept]. Like the definition, SEP not only maintains the P2PSIP overlay topology, but also provides distributed database service.

1. SEP uses a flexible packet forwarding mechanism so that peers could choose the best peer to route the packet further. For example, if a peer selects a downstream peer which has no enough resource at that time, then the requests may be discarded by the downstream peer. So the upstream peer could choose another downstream peer which is in good condition.

2. SEP provides a relatively general method to discover which peers could provide a specific service, for example, STUN service or STUN Relay service. This kind of information is needed by other peers, otherwise these peers won’t be able to complete some tasks.

3. The routing modes taken by the SEP attempts to make the transaction with lower latency and higher success rate even if intermediate peers fail simultaneously or NATs exist between the source peer and the destination peer.

2. Overview

Each peer in the overlay keeps a few routing states, including routing table, neighbor table or other states which are maintained by the DHT algorithms. According to the DHT algorithms, the peer must learn other peers’ information and collaborate with them. So the routing states not only keep the network reachability information of the next hop peers, but could be extended to store other information, for example, service capability, processing status, etc.

Peers with heterogeneous capability make up of the P2PSIP overlay. SEP attempts to benefit from the heterogeneity among the peers and mitigates the bad effect resulting from it at the same time. Some peers could provide other service in addition to the routing and storage service which are the required services for peers. These additional services include STUN, STUN Relay, etc, which are needed by other peers, for example, which are behind the NAT. In terms of processing resource, such as bandwidth, CPU cycles, they are also heterogeneous. This kind of heterogeneity may make the less powerful peer congested by the messages from the more powerful peers and fail the transactions processed by the congested peers. In this case, a simple flow control mechanism between immediate peers combined with a flexible forwarding mechanism would make the situation better.
The communication between peers may not work in the presence of the NAT between them. There are a suite of proposals STUN/TURN/ICE in IETF to address this problem. SEP works with these proposals and provides a built-in NAT traversal functionality.

2.1. Routing States

Each peer should maintain routing states and route packets by choosing the next hop peer from the routing states. The routing states should be updated as the overlay topology changes. Here, we give an example data organization of routing states, and explain each item in the entry. Although this is an implementation issue, comprehension on the routing states will make reader understand SEP easier.

In current DHT algorithms, there are often two kinds of routing states: routing table and neighbor table. Routing table is often asymmetric and neighbor table is symmetric. Peers in the routing states are also called P2PSIP neighbors defined in the [concept]. It means the peer could reach these peers directly without further lookup.

The routing states are often comprised of several entries. Each entry keeps the information about the P2PSIP neighbors. The information about the P2PSIP neighbors might be organized as follows.

- **Peer-ID:** which uniquely identifies a peer in the overlay.
- **Workable candidate pairs:** this item records several candidate pairs which are got by using ICE procedure. By using them, peers could communicate with its neighbors directly. Two or more workable candidate pairs could improve the stability of the P2PSIP peer control connection.
- **Current processing status:** this item reflects the processing status of the peer’s neighbors. Due to heterogeneity in peers’ processing power and traffic load, some peers may be overloaded with excess message. So it is useful for the peer knows the processing status of its neighbors. The peer may evaluate the downstream peer’s status during the routing decision process. So that the flexible forwarding mechanism could be used to choose the healthier peer among its neighbors. The possible states could be classified into normal or busy. They also could be measured by some quantitative parameters, such as CPU idle percentage, available bandwidth, etc.
- **Service capability:** it gives which additional services the peer supports. Having this kind of information in mind, the peer learns which neighbors could provide a specific service and provide this.
kind of information to other peers who need the specific service.

Note: here, we just list some basic items of the information about the P2PSIP neighbors. This information is not exhaustive and it will be extended to support new functions if needed.

2.2. Data Operations

A peer may put multiple resource objects in the overlay under the same resource ID, and some peers may put various resource objects with the same resource ID. Therefore, there may be multiple resource objects under the same resource ID. SEP uses the owner identity and hash code accompanied with resource ID to locate the unique resource object.

A resource object in the P2P overlay network can be any type of data, such as a contact address, a file, etc. The size varies from one resource object to another. Resource objects with small size can directly be put to or got from the overlay by carrying them in the PUT or GET message. If the size is over some limit, it’s more efficient to exchange the resource object using a direct connection. SEP supports both kinds of operation. Negotiation information is put into the PUT or GET message to negotiate a direct data transport for the latter.

The resource objects should be transferred from one peer to another when the overlay churns. But the sending peer may not be aware of the state of the receiving peer, and whether the receiving peer is really responsible for the <key, value> pairs to be transferred. SEP provides a TRANSFER operation with negotiation capability to solve these problems.

2.3. Data Transfer During the Overlay Churn

In P2P overlay, peers can join and leave the overlay at any time. Therefore two operations of data moving and routing states update should be performed accordingly. During the churn, some data may be placed on the wrong peer because the two operations can’t keep pace with each other. In the meantime, a lookup operation for a key may fail in that the data associated with the key is not on the correct root peer.

SEP attempts to reduce the failure possibility of the transaction during the overlay churn. The idea is to make the joining peer or leaving peer work together with its neighbors to serve the coming lookup operations. For example, when a peer is about to leave the network, it could transfer <key, value> pairs to its neighbors and notify other peers its departure simultaneously.
The leaving peer and its neighbors may be the target of the lookup operation due to routing states update in other peers. So they could serve the lookup operation together by leading the requests to which if they have no answer to the others.

3. 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 [1].

Some of the terminology has been borrowed from the [concept].

- Routing states: it is a general description of the information used to route packets through the overlay. Several kinds of routing states exist. Routing table and neighbor table are two examples.

- Service peer: peers who provide one or more specific services other than routing and storage service.

- Public neighbor peer: refers to the peers who are on the public Internet and also the P2PSIP neighbors.

- Upstream peer/downstream peer: They are paired with each other. The distance between the upstream peer and the downstream peer is only one hop in the overlay. Downstream peer appears in the routing states of the upstream peer, so the downstream peer often receive the P2PSIP control packet from the upstream peer. Downstream peer is often the neighbor of the upstream peer and appears in its routing state.

- Transaction: a transaction is initiated by the source peer and comprises a request and several responses. Transaction is an end-to-end concept and is uniquely identified by the tuple, including source peer-ID, destination ID, message type, transaction ID (chosen by the source peer).

- Relaying peer: Peers which have direct connections with the source peer and are willing to relay packets destined to the source peer. Relaying peers MUST be on the public Internet and used while the destination peer has trouble in sending response to the source peer behind NAT.

4. Design Choice

In this section, some considerations about the SEP’s design are
presented. The principle behind the design is to make the transaction with shorter latency and higher success rate in the environment where peers churn frequently and some peers may be overloaded by traffic. So we choose the semi-recursive routing mode combined with overlay native routing mode and propose a flexible forwarding mechanism to avoid the congestion in the overlay. Other design choices such as end-to-end reliability and service discovery are also discussed in this section.

4.1. Routing Mode

There are several routing modes used to realize the P2PSIP transaction. As for P2PSIP transaction, requests have to be forwarded through the overlay by using the overlay routing states, but the responses may take several ways to go back to the source peer. First, the destination peer could send the responses to the source peer along the reverse path which requests have traversed. The second one is for the destination peer to route the responses directly to the source peer by routing in the IP layer. The last one is still to use overlay native routing to return the responses.

In the following figure, we summarize these four routing modes and compare them from three perspectives: difficulty of NAT traversal, resilient from the failure of intermediate peers and the latency of the transaction.

<table>
<thead>
<tr>
<th>Routing Mode</th>
<th>Difficulty of NAT traversal</th>
<th>Resilient from the failure of intermediate peers</th>
<th>Latency of the transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>iterative</td>
<td>high</td>
<td>high</td>
<td>high</td>
</tr>
<tr>
<td>recursive</td>
<td>low</td>
<td>low</td>
<td>medium</td>
</tr>
<tr>
<td>semi-recursive</td>
<td>medium</td>
<td>high</td>
<td>low</td>
</tr>
<tr>
<td>overlay native</td>
<td>low</td>
<td>high</td>
<td>high</td>
</tr>
</tbody>
</table>

Figure 1 the comparison of the four routing modes

For iterative mode, NAT traversal is a big problem and the transaction latency is high. Recursive mode has little trouble in traversing NAT, but transaction would fail if any one of the intermediate peers in the path fails. As for overlay native mode, the response is still routed to the source peer through the overlay and hence the transaction’s latency is high. Semi-recursive mode has good balance between the three perspectives. The response is returned to the source peer directly. The only problem in the semi-recursive mode is that the response may not reach the source peer due
to the existence of NAT.

SEP takes the two routing modes from them, semi-recursive mode and overlay native routing mode. In most cases, semi-recursive mode works. Overlay native mode may be used once semi-recursive mode has failed. As for NAT traversal in semi-recursive mode, SEP’s solution is for the source peer to convey some relaying peers to the destination peer so that the destination peer could send the response to these peers relaying the response to the source peer. Of course, if the source peer is on the public Internet, the destination peer could reach the source peer directly without any help.

4.2. Service Discovery

Heterogeneity in peer’s capability signifies that some peers provide not only the required service for the peer, but also some additional services which may be very important for other peers. STUN and STUN Relay are two example services.

The problem is how the peers in need of this service to discover these service peers. SEP defines a message, named LookUpServicePeer, to discover them with the help from other peers while the message is routed through the overlay. As described in the section 2.1, every peer keeps its neighbors’ service capability information. When the peers receive the LookUpServicePeer message, it could put the service peers what it knows into the request, and the source peer who sends the LookUpServicePeer message will get them from the response in the end.

Due to service peers’ distribution, this method could not ensure to get service peers for certain in one LookUpServicePeer transaction. But the peers in need of a specific service could try this lookup operation several times and choose different key to make different message cover as much peers as possible.

This service discovery method is so general that it could apply to the discovery of any kind of service. It could be used to discover STUN server, STUN Relay server and other service peer.

4.3. Processing Status Advertisement from the Downstream Peers

Downstream peers should advertise its processing status to its upstream peers in order to prevent the packets from being forwarded to a busy downstream peer. The advertisement should be done periodically to reflect the real time status as much as possible.

The processing status advertisement may be conveyed in the periodic keepalive messages. But if there are substantial changes in the
peers’ processing status, it may send a notification immediately.

Open issue: how do we define the processing status of a peer? Is it in terms of the available bandwidth, CPU cycles or some other parameters? The second question is: when will we think the peer has experienced a substantial change in the processing status?

4.4. An flexible Forwarding Mechanism

Here, we will introduce the possible congestion in the overlay and propose a flexible forwarding mechanism in the overlay to mitigate the effects from the congestion.

Because of the heterogeneity in peers’ bandwidth, processing power and traffic load, some less capable peers may be overloaded by excess messages. The messages may be discarded by these peers. Fortunately, there are a great number of paths between the source peer and the destination peer in the overlay. So if the upstream peer learns that the downstream peer is in the busy state, the upstream peer could choose another one which is in good condition so long as the choice complies with the DHT algorithm in use. But it may experience longer latency when this flexible forwarding operation is taken.

At least two advantages could be got from the use of the flexible forwarding mechanism. One is to divert the traffic from congested peers to others and the other one is to improve the success rate of the transaction in the overlay as much as possible.

4.5. End-to-end Reliability

Reliability is very important for the transaction in the peer protocol. The transaction comprises of requests and responses. Because peer protocol messages will be delivered to the destination hop-by-hop, it is hard to achieve end-to-end reliability for the transaction. The reason for this is that source peer does where the destination peer is before the request really reaches the destination. So it could not rely on TCP to ensure end-to-end reliability. Although every hop in the path could be TCP connections, it only guarantees the reliability between the immediate peers. The behavior of the intermediate peers may break the end-to-end reliability, for example, dropping packets.

It seems that application retransmission mechanism is the only answer to the end-to-end reliability. If the source peer could not receive the response within the expected time, it could retransmit the request. It works for the simple request-response communication.
Open issue 1: how does the source peer set the retransmission timeout timer?

Open issue 2: if we take the simple request-response communication mode, retransmitting request works. But NAT traversal is a challenge in the P2P case, so for the message used to negotiate the ICE parameters, only retransmitting request may not work. It is because ICE needs to perform connectivity checks after the transaction ends. For the destination peer, it is not sure whether the response has returned to the source peer and when it could start the ICE procedure. What will the destination do if the response failed?

5. Message Syntax

SEP protocol messages comprises of two parts: message header and some attributes expressed in a TLV style.

O Message header: it contains some information for forwarding operations, for example, destination ID, message type and some options; and other information used to for the source peer to match the response with the request, including source peer ID, destination ID, transaction ID.

O Attributes: in order to make SEP extensible, TLV-style attributes is used to express attributes. In order to support new operations, we are not only able to add new messages, but able to add new attributes. Every attribute may be composed of other attributes with the TLV-style.

5.1. Message Header
Version (4 bit): Indicates the version of the SEP protocol. The current version is 0x01;

R (1 bit): If it is set, used to indicate that this message is a response.

H (1 bit): If it is set, used to indicate to the intermediate peers that this message should be processed in terms of the message type in addition to forwarding operation.

D (1 bit): If it is set, it indicates to the destination peer that the source peer thinks that a direct connection SHOULD be negotiated and established after this transaction.

F (1 bit): If it is set, it means the response should be processed in the relay mode.

J (1 bit): If it is set, it means that it has been processed only by the source peer.

Message type (8 bit): type of the message.

TTL (8 bit): time-to-live. It indicates the number of hops which a message is allowed to traverse before it is discarded.

Message Length (16 bit): indicates the length of the message, including the message header and the following variable attributes.

Transaction-ID (32 bit): It is randomly assigned by the source peer and in order to match the response with the request.

Source Peer-ID (128 bit): indicates the Peer-ID of the source peer who initiates the request.
Destination ID (128 bit): either a destination Peer-ID or a destination key. Its function is for the P2P overlay to get the packet delivered to the destination peer.

5.2. Message Attribute

In this draft, we define several message attributes that are expressed in a TLV-style.

5.2.1. TLV Format

0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| M | Reserved   |     Type      |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+  Value - variable length                                      +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

M (1 bit): It indicates that the attribute MUST be understood by the peers who intend to process them. If the attribute could not be understood by the processing peers, the message MUST be discarded or return a error response.

Type (8 bit): It indicates the type of the attribute.

Length (16 bit): the byte length of the attribute.

5.2.2. Peer Address Info

Attribute type: 0x00 (to be assigned by IANA)
Trans (4 bit): it indicates the type of the transport protocol. In this draft, we define two types: 0x00(TCP), 0x01(UDP).

Type (4 bit): host (0x0), server reflexive (0x01), peer reflexive (0x02), relayed address (0x03). The classification is the same as that in [ICE].

Port (16bit): the port on which this peer listens for requests.

Priority (32bit): It is used by ICE procedure to sort the candidate pair. It is computed as suggested in [ICE].

IPv4 Address: the IPv4 address of the peer.

5.2.3. Peer service capability

Attribute type: 0x01 (to be assigned by IANA)

This attribute is to express which services the peer could provide to the other peers. Every bit in the value is used to indicate a specific service. In this draft, we recommend the length of this attribute is 16 bit long and three services are defined.

N (1 bit): if N flag is set, it means the peer is behind NAT. Otherwise the peer is on the public Internet.

S (1 bit): STUN service. If it is set, it means the peer supports STUN service.
T (1 bit): STUN relay service. When it is set, it means the peer supports STUN relay service.

### 5.2.4. Peer-ID

Attribute type: 0x02 (to be assigned by IANA)

```
+-------+--------+--------+--------+--------+--------+--------+--------+--------+--------+
| M     | Reserved| Type   | Length | Peer-ID = 128bit |
+-------+--------+--------+--------+------------------+
```

Peer-ID conveys the Peer-ID of the specific peer. The length of this field is 128 bit. Although not all DHT algorithms use 128 bit as the length of the Peer-ID, we think that 128 bit works for all DHT algorithms.

### 5.2.5. Source Peer Info

Source Peer Info is a composite attributes. It is comprised of Peer-ID attribute, peer service capability attribute and several peer address info attributes. This attribute carries a few source peer related information to other peers.

Attribute type: 0x03 (to be assigned by IANA)

```
+-------+--------+--------+--------+--------+--------+--------+--------+--------+--------+
| M     | Reserved| Type   | Length | Peer-ID | Peer service capability | Peer Address Info - 1 | Peer Address Info - N |
+-------+--------+--------+--------+--------+-------------------------+-----------------------+-----------------------+
```
5.2.6. Destination Peer Info

Destination Peer Info is also a composite attribute. Like the source peer info attribute, it is also comprised of Peer-ID attribute, peer service capability attribute and several peer address info attributes. This attribute carries a few destination peer related information to the source peer.

Attribute type: 0x04 (to be assigned by IANA)

```
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   M  |  Reserved               |    Type      |     Length               |
+-----------------------------------------------+
|                 Peer-ID                           |
+-----------------------------------------------+
|                Peer service capability           |
+-----------------------------------------------+
|                Peer Address Info - 1             |
|                   ............                  |
|                Peer Address Info - N             |
+-----------------------------------------------+
```

5.2.7. Resource Info

Attribute type: 0x05 (to be assigned by IANA)
<table>
<thead>
<tr>
<th>M</th>
<th>Reserved</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>O</td>
<td>H</td>
<td>C</td>
</tr>
<tr>
<td>+ Resource ID(16Bytes) +</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ Expiration Time(4Bytes,Optional) +</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ Owner Identity(16Bytes,Optional) +</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ Hash Code(16Bytes,Optional) +</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ Value Content(variable length,Optional) +</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

E (1 bit): If this bit is set, Expiration Time field MUST be included; otherwise not.

O (1 bit): If this bit is set, Owner Identity field MUST be included; otherwise not.

H (1 bit): If this bit is set, Hash Code field MUST be included; otherwise not.

C (1 bit): If this bit is set, Value Content field MUST be included; otherwise not.

Value Type: A 12-bit value, the type of application in which the value is used. How to assign this value depends on the content to be stored in the overlay. It will be developed in the later version.

Value length: The number of Bytes in the value.

Resource ID: A 16-byte value, the hash function result of the resource unique property.

Expiration Time: Optional in the Resource Info, A 4-byte value
indicate the expiration time of the resource object.

Owner Identity: it specifies the owner of the Resource object.

Hash Code: It is computed by hashing the resource objects. It is used for integrity check. A resource object must be put into the overlay with its owner identity and the hash code of the resource object. The owner uses the owner identity and the hash code to refresh and modify the unique resource object, because a user may put several resource objects under the same resource ID.

Value Content: when presented, its length is specified by the "Value Length" parameter.

5.2.8. Negotiation Info

Attribute type: 0x06 (to be assigned by IANA)

This attribute is used to negotiate the transport parameters for PUT or GET or TRANSFER operations. Its format is to be determined.

5.2.9. Response Attribute

Attribute type: 0x07 (to be assigned by IANA)

0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|M|  Reserved   |     Type      |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Response code          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

The following response codes are defined in this version.

200: Successful request;

401: Bad parameters;

402: Could not find the corresponding data;

403: The request is rejected for some reason;

5.2.10. Service Peer Info

Attribute type: 0x08 (to be assigned by IANA)
It is also a composite attribute. It intends to collect the reachability information of peers which could support the service specified in the attribute.

- **Num of Peers (8 bit)**: this field records how many peers in this attribute. It is set to zero while the source peer creates this attribute. The number will be modified by the intermediate peers and destination peers when new peers are added. This attribute should be returned to the source peer in the response from the destination peer.

- **Service type (8 bit)**: this field indicates which type of service the source peer wants to know. In this draft, three service types are recommended. They are STUN (0x00), STUN relay (0x01) and Public peer (0x02).

- **Max. Num (8 bit)**: this field indicates the maximum number of the service peers what the source peer wants.

- **Peer-ID attribute**: this field has a type of Peer-ID attribute and records the Peer-ID information of the peers who provides the specified type of service.

- **Peer address info attribute**: this field has a type of Peer Address Info attribute and records the transport address of the peers who could provide the specified type of service.
5.2.11. Relaying Peer Info

Attribute type: 0x09 (to be assigned by IANA)

```
+--------------------------------+---------------------------------+--------------------------------+---------------------------------+
|M|  Reserved   |     Type      |           Length              |
+--------------------------------+---------------------------------+--------------------------------+---------------------------------+
| Num of Peers |              Reserved                         |
+--------------------------------+---------------------------------+---------------------------------+
| Peer Address Info attribute 1 |
+--------------------------------+---------------------------------+---------------------------------+
|                                |
+--------------------------------+---------------------------------+---------------------------------+
| Peer Address Info attribute N |
+--------------------------------+---------------------------------+---------------------------------+
```

This attribute intends to convey relaying peer info to the destination peer which could return response to the source peer with the help from these peers in the presence of NATs between them.

O Num of Peers: this field records the number of relaying peers supplied by the source peer. It won’t be changed in transit.

O Peer Address Info attribute: this field has a type of the Peer Address Info attribute and records the reachability information of the relaying peers.

5.2.12. Source Reflexive Address attribute

Attribute type: 0x0A (to be assigned by IANA)

```
+--------------------------------+--------------------------------+--------------------------------+---------------------------------+
|M|  Reserved   |     Type      |           Length              |
+--------------------------------+--------------------------------+--------------------------------+---------------------------------+
| Port             | Transport pro.|   Reserved    |
+--------------------------------+--------------------------------+---------------------------------+
| IPv4 Address                                             |
+--------------------------------+--------------------------------+---------------------------------+
```

This attribute is often used in the case where bootstrap peer processes the JOIN request and record the source transport address of the JOIN request including IP address and the port observed by
bootstrap peer. This attribute will be delivered in the request and later in response unmodified. When the Join response returns to the bootstrap peer, it will check this attribute and make it as the destination transport address of the JOIN response relayed by the bootstrap peer. In this way, bootstrap peer won’t keep state for the joining peer and make the response return to the Joining peer even though the Joining peer is behind NAT.

O Port (16 bit): the port number observed by the bootstrap peer.

O Transport Pro.: The type of transport protocol. In this draft, two types are defined. TCP (0x00) and UDP (0x01).

O IPv4 address: the IP address observed by the bootstrap peer.

5.2.13. Routing table

Attribute type: 0x0B (to be assigned by IANA)

The attribute format depends on detailed DHT algorithm, so its format is TBD.

5.2.14. Status Info

Attribute type: 0x0C (to be assigned by IANA)

```
+-----------------+--------------+---------------+-----------------+
| M | Reserved | Type | Length |
+-----------------+--------------+---------------+-----------------+
|         | Congestion State |         |
+-----------------+-----------------+
```

Congestion State: indicates the current state of a peer. 0x00, a peer which states the peer is in good condition; 0x01, which means the peer is busy.

5.2.15. Event Info

Attribute type: 0x0D (to be assigned by IANA)
5.2.16. Update Type

Attribute type: 0x0E (to be assigned by IANA)

<table>
<thead>
<tr>
<th>M</th>
<th>Reserved</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</td>
<td>0 1 2 3 4 5 6 7 8 9 0 1</td>
<td></td>
</tr>
</tbody>
</table>

Update Type: indicate what kind of Update operation the sending peer want the destination peer do. 0x00 means the receiving peer should check whether it should update its routing state; 0x01 means the sending peer requests routing table of the receiving peer.

6. Peer Protocol Message

6.1. Overlay Maintenance

6.1.1. JOIN

A peer MUST learn some necessary information before it wants to join the overlay, such as DHT algorithm, bootstrap peer list, its Peer ID, hash algorithm used to get the key and the like. It could get that information by enrollment procedure or some other means which is not within the scope of this draft. In SEP, the bootstrap peer MUST be on the public Internet.

The joining peer sends JOIN request to the bootstrap peer. In the message header, the joining peer SHOULD fill Source Peer ID and Destination ID with its own Peer ID. If the joining peer is behind
NAT, it SHOULD set D flag in the message header to indicate to the
destination peer that they should start an ICE negotiation with each
other to find a direct connection for later communication. Source
Peer Info MUST be carried in the JOIN request.

Join request =
  Message Header
  Source Peer Info
  [Relaying Peer Info]
  [Source Reflexive Address]

When a peer receives JOIN request, it checks the Destination ID to
determine whether it is the destination of the join request. If it is
not, the peer forwards the message to next peer which is closer to
the destination. If the peer is the destination of the JOIN request,
the peer processes the message and sends JOIN response to the joining
peer. The destination peer often sends its routing table to the
joining peer in the response. If the destination peer learns that
source peer is behind NAT by check the Source Peer Info, it MUST set
the D flag in the response.

Join response = Message Header Response Attribute Destination Peer
  Info [Source Reflexive Address] [Routing Table]

Join response =
  Message Header
  Response Attribute
  Destination Peer Info
  [Source Reflexive Address]
  [Routing Table]

In the end, the joining peer receives the response. If it is a 200
OK response, the joining peer MAY use the routing table in the
response to construct its own routing table. If the D flag is set in
the response, the joining peer SHOULD start ICE procedure to find out
a direct path between them.

JOIN operation will trigger data transfer from other peers to the
joining peer. These peers who will transfer data to the joining
peers are often the neighbors of the joining peer. After the
transfer finishes, these neighbors will update their routing table to
reflect the existence of the joining peer.

6.1.2. LEAVE

Before it leaves the overlay, a peer should transfer all the data to
other peers, and other peers should update their routing table. So
the leave process MAY include the following steps:
1. The leaving peer sends LEAVE request to its neighbor peer(s);
2. the neighbor peer(s) expands the key region and updates the routing table;
3. the leaving peer notifies its upstream peers its departure;
4. the leaving peer starts data transfer process;
5. the leaving peer and the neighbor peer work together to serve the new requests;
6. After data transfer finishes, the leaving peer leaves the overlay;

A leaving peer sends leave message to its neighbor peers which depend on the DHT algorithms in use.

Leave request =
   Message Header
   Source Peer Info

The peer receiving the LEAVE message SHOULD consider whether its responsible space in the overlay has changed due to the source peer’s leave and then expand the responsible space accordingly. If the neighbor peer receives message, the peer MAY either reply this request on its own if the associated data has been transferred from the leaving peer, or lead this request to the leaving peer.

Leave response =
   Message Header
   Response Attribute
   Destination Peer Info

When the leaving peer receives the response, it could start data transfer operation to its neighbors and at the same time, it could notify any of its upstream peers its departure to accelerate the convergence of the overlay. Before the transfer finishes, the leaving peer and its neighbors could work together to serve the requests in the overlay. (TO DO: how they work together with various messages is to be developed further.)

Open issue: How do the leaving peer and the neighbor peer work together to serve the new request?

6.1.3. KEEPALIVE

KEEPALIVE message is used by a peer to make sure whether the neighbor peers are still alive. If a peer has not received KEEPALIVE response from the destination peer for several times, it could conclude that the destination peer is not alive and then should start to update its routing or neighbor table. After receiving KEEPALIVE request, the destination peer MUST convey its status information in the KEEPALIVE
response. If a peer has received a KEEPALIVE message with a peer’s current status which shows this peer is in the busy state, the peer SHOULD not forward packets to that congested peer until that peer reaches the normal state again.

KEEPALIVE Request =
Message Header
Source Peer Info
(preamble)
KEEPALIVE Response =
Message Header
Response Attribute
Destination Peer Info
Status info
(postamble)

6.1.4. NOTIFY

The continuity and quality of service contributed by a peer is concerned by the other peers in the overlay. After receiving an indication that a specified peer is interested in the status of the contributed service, the peer who contributed the service starts to monitor its service status for the specified peer, and then it informs the specified peer about the interesting service status information through a NOTIFY message when it finds that the service status changes (e.g. service quality degraded or service interrupted). Those information includes: a) an event means that the peer will leave; b) an event means that the peer is coming into congestion state, etc.

NOTIFY message =
Message Header
Source Peer Info
[EVENT] [STATUS INFO]

When receiving notify message, peer MAY choose to update corresponding status according to the information in the request.

Open Issue: As proposed in the section 4.4, current status advertisement from downstream peers will be useful to ease the congestion of the downstream peers. How do these downstream peers measure its status? When is the right time for them to advertise the status change?
6.1.5. UPDATE

UPDATE message is used to notify its existence in the overlay to its neighbors or request routing table from its neighbors. The target peers of the UPDATE request depends on the DHT algorithms. For example, neighbor may just mean the successor or predecessor in Chord, but it means all peers in its leaf set in Pastry.

UPDATE provides two options for peers to update its own or its neighbors routing states. One option is for the peer to notify its existence to its neighbors. For example, after the joining peer finished JOIN transaction in Pastry, it should notify other neighbors to let them update its leaf set. The other option is for the peer to request routing table from its neighbors. In Pastry, peers may want to update its routing states after it checks some peers have been dead. They often request routing table from its neighbors to achieve this. Of course, SEP could support the two options in the single UPDATE transaction.

Before UPDATE request is sent, the source peer should set the UPDATE Type attribute to determine option type.

When the destination peer receives the UPDATE request, it first checks the UPDATE Type. If the UPDATE is for a routing table, the destination peer should convey the routing table according to the request in the response.

**UPDATE Request**

```
Message Header
Source Peer Info
Update Type
```

**UPDATE Response**

```
Message Header
Response Attribute
Destination Peer Info
Update Type
[Routing-table]
```

6.1.6. SearchPeer

One Peer sends a SearchPeer request to find the responsible peer for a destination ID. The destination peer information MAY be used to maintain the overlay routing table. The source peer MAY establish a direct connection with the destination peer. If it is the case, the D flag MUST be set to indicate to the responsible peer they SHOULD exchange ICE candidate for the NAT traversal. In Chord, one peer uses SearchPeer request to lookup the responsible peer for its finger
table. The responsible peer MUST make a response to the request with its own peer information. Current status of the responsible peer MAY also be included in the response.

SearchPeer Request =
  Message Header
  Source Peer Info
  [Relaying Peer Info]

SearchPeer Response =
  Message Header
  Response Attribute
  Destination Peer Info
  [Status Info]

6.1.7. TRANSFER

When a peer joins or leaves the P2P overlay network, data transfer operation will be triggered. The joining peer will be responsible for some <key, value> pairs and the leaving node SHOULD transfer its stored <key, value> pairs to its neighbors. When the transfer operation takes place, the source peer does not know whether the peer receiving the <key, value> pair is leaving the overlay or it is busy. So SEP provides a mechanism to make both peers could negotiate some transfer parameters and the time when will the transfer start.

The source peer sends a TRANSFER request with the negotiation of the range of the resource ID to be transferred and the destination peer responds with the negotiation result by taking its current state into account. If the destination peer is about to leave the overlay, it MAY reject the request and suggest the source peer tries another peer some time later. The destination MAY accept, redirect or deny the transfer operation. After that, the source peer transfers resource objects to the destination peer if the request was accepted. Peers MAY also negotiate transport parameters, such as transport protocol, to transfer resource objects.

TRANSFER Request =
  Message Header
  Source Peer Info
  [Negotiation Info]
  [Resource Info]
TRANSFER Response =
    Message Header
    Destination Peer Info
    [Negotiation Info]

6.2. Data Operations

6.2.1. PUT

A peer sends a PUT request to publish its resource object in the P2P overlay. A peer may also send a PUT request to refresh or modify the resource object. A hash code must be used with the owner identity to identify the unique resource object under the same resource ID. Every resource object has an owner identity to verify the owner of the resource object and a hash code to check the integrity of the resource object. Every resource object also has an expiration time. One peer can refresh the resource object by modifying the expiration time, otherwise the destination peer SHOULD remove the resource object when it expires.

A PUT request may take the resource object within the request. A PUT request may also include negotiation information to negotiate the transport parameter to set up a direct connection between the source peer and the destination peer, and then the resource object transports through a direct connection. When the resource object transport completes, the source peer may notify the destination peer of the completion. When the put request is used to refresh the resource object, there is no resource object in the request but the resource ID, expiration time, owner identity and hash code are needed.

PUT Request =
    Message Header
    Source Peer Info
    [Relaying Peer Info]
    Resource Info
    [Negotiation Info]

PUT Response =
    Message Header
    Response Attribute
    Destination Peer Info
    [Negotiation Info]

6.2.2. GET

The source peer sends a GET message with the resource ID and content type to retrieve the resource object. The resource objects can be
sent back with the Get response. If the size of the resource objects is large (over some limit), the transport parameters can be negotiated using the GET request and response to figure out the transport type of the resource objects in a following direct connection. When the resource objects transport completes, the source peer must notify the destination peer of the completion.

GET Request =
Message Header
Source Peer Info
[Relaying Peer Info]
Resource Info
[Negotiation Info]

GET Response =
Message Header
Response Attribute
Destination Peer Info
[Resource Info] [Negotiation Info]

6.2.3. REMOVE

A peer sends a REMOVE request to delete the resource object in the P2P overlay network. A resource ID is used to locate in which peer the resource object is stored. The owner identity and the hash code is used to distinguish the resource object from others under the same resource ID. One responsible peer should delete all information about the resource object including replicas immediately when it receives the remove request.

REMOVE Request =
Message Header
Source Peer Info
[Relaying Peer Info]
Resource Info

REMOVE Response =
Message Header
Response Attribute
Destination Peer Info

6.3. Additional messages

6.3.1. LookUpServicePeer

Some messages could be used to discover service peers who could provide a specific service, for example, STUN or STURN Relay service. Source peer MUST carry Service Peer Info in the request. In SEP, we
choose the LookUpServicePeer to collect the service peer’s information. This message will be forwarded through the overlay in a hop-by-hop way, so intermediate peers and the destination peer could collect the service peers based on its local knowledge of the overlay. Every peer other than the source peer SHOULD put the searched results in the Service Peer Info attributes which will be returned back to the source peer.

The source peer MUST insert a Service Peer Info attribute in the request to collect service peers whose types may be STUN, STUN relay, etc, and send it to the destination peer by choosing a given destination ID. The source peer should specify the maximum number of service peers in the request. If the number of peers reaches the maximum number, the successive intermediate peer or destination peer won’t need to find satisfied peers. In the resulting Service Peer Info attribute, none of the peer has the same Peer-ID attribute. If the successive intermediate peer or the destination peer finds the service peer has already been in the Service Peer Info attribute, they MUST not be put into the Service Peer info attribute.

When the peers process LookUpServicePeer request or response, some message-specific actions should be done in addition to basic actions proposed in section 9.

The source peer SHOULD choose a destination ID which will determine the path the request will traverse. The source peer SHOULD set the H flags in the message header and put in the request a Service Peer Info attribute which SHOULD set the maximum number of the expected service peers. When source peer receives the response, it should check whether Service Peer Info attribute exists. If it is, the source peer gets service peers from it for later use.

While intermediate peers and the destination peer receive the LookUpServicePeer request, it first checks if the number in the Service Peer attribute reaches the maximum value. If it is not the case, the intermediate peer should get the service type what the source peer wants and try to search the satisfied peers in the locally stored states. If the intermediate peer gets some service peers, it should put it in the Service Peer info attribute. If the number of the discovered peers equals the maximum number, they will not be put in the attribute. At the same time, every intermediate peer or the destination peer should guarantee that every service peer is unique in this set.

When the destination peer generates the response, it should carry the Service Peer attribute which is copied from the request unmodified. In the end, the discovered service peers are learned by the source peer by carrying them in the Service Peer Info attribute.
LookUpServicePeer Request =
  Message Header
  Source Peer Info
  Service Peer Info
  [Relaying Peer Info]

LookUpServicePeer Response =
  Message Header
  Source Peer Info
  [Service Peer Info]

7. Forwarding Operations

Forwarding operation is the basic and the important function of the peer in the overlay. Peers should choose the next hop peer for the packet regardless of where packets are from. The packets as the input to the forwarding operation may come from the upper applications in the same peer, or from the network. But the general forwarding operation applies to all of them.

SEP protocol takes the semi-recursive and overlay native routing modes by considering the difficulty of NAT traversal and simultaneous failures of the intermediate peers.

7.1. Request Forwarding

If the packet is a request, it should be routed through the overlay by looking up the downstream peer based on the destination-ID in the packet. The destination ID may be a Resource-ID or a Peer-ID which depends on the message type.

In order to improve the reliability of the packets, SEP makes a flexible forwarding decision, not only complying with the DHT algorithms, but also based on the processing status of the downstream peers. The flexible forwarding operations may introduce latency for the packet, but it does improve the reliability. Certainly, the tradeoff between the reliability and the additional latency from the additional hops should be evaluated independently by the forwarding peer.

Note: if there are several candidate downstream peers at the same level in the overlay, for example, peers in the successor list in chord, the flexible forwarding operation would not introduce additional significant latency.

The forwarding rules for the request are listed as follows:
1. Get the destination ID from the request;
2. Check whether the peer itself is responsible for the destination ID. If it is, then deliver the packet up to the message specific function;
3. If it is not, get a few downstream peers closer to the destination ID in the routing states and get their associated processing status;
4. It is up to the implementation to decide which peer the request will be sent to by evaluating the trade-off between the reliability and the latency;

7.2. Response Forwarding

If the packet is a response, the first choice is to send the response back to the source peer directly, because the destination peer could get IP address of the peer from the Peer Address Info attribute. Due to the existence of the NAT, response MAY be returned to source peer by using some special mechanisms. The forwarding behavior on intermediate peers or destination peer is different which are described below respectively.

7.2.1. Response Forwarding on the Destination Peer

1. Before the destination peer sends the response back to the source peer, it should check the N flag of Service Capability Info to learn whether the source peer is on the public Internet;
2. If the source peer is on the public Internet, the transport address in the Peer address Info could be used to send the response back directly.
3. If the source peer is behind the NAT, the destination peer checks whether the Relaying Peer Info attribute is carried in the request. If the destination peer has not found them, the response should be sent back by routing through the overlay and F flag in the message header MUST be cleared.
4. If the Relaying Peer Info is carried in the request, then destination peer could send the response to the transport address of these relaying peers which will relay the response to the source peer and the F flag MUST be set. Note: destination peer could send the response to these relaying peers parallelly or sequentially. The response could also be returned by using overlay native routing. It is up to the destination peer to choose the right ways to send the response back. Open issue: How could the destination peer make sure that the response has reached the source peer?
7.2.2. Response Forwarding on the Intermediate Peer

1. The intermediate peers check the F flag in the message header first. If it is cleared, the response SHOULD be treated via overlay routing according to the forwarding rule described in the section 8.1.
2. If the F flag is set, it means that the response want to be relayed by the intermediate peer to the source peer. So the intermediate peer gets the source peer-ID and checks whether the peer has the direct connection with the source peer.
3. If the direct connection exists, the intermediate peer sends the response to the source peer through the established connection with it.
4. If there is no direct connection between them, then it will check whether Source Reflexive attribute is carried in the response. If the Source Reflexive attribute exists, the intermediate peer will send the response directly to the transport address recorded in the Source Reflexive attribute;
5. If it does not exist, the intermediate peer discards the response silently.

8. General Peer Behavior

A few kinds of peers are involved in the transaction, including source peer, destination peer, intermediate peers. The request is generated by the source peer and then forwarded through the overlay by traversing several intermediate peers and reaches the destination peer in the end. The response from the destination peer may either be sent via IP routing, or forwarded by the intermediate peers through the overlay. In this section, we give the general behavior of the peers who plays different roles in the transaction.

8.1. Source Peer Behavior

8.1.1. Generating the Request and Sending the Request

The fields in the message header and some required attributes of the new message MUST be determined before the message would be sent.

O Version: the current version number is 1;
O F flag: this bit MUST be cleared in the request;
O D flag: If the source peer realizes that it is behind NAT and needs a direct connection with destination peer after this transaction completes, the D flag MUST be set.
O H flag: If the source peer wants the intermediate peers to process this message hop-by-hop, the H flag MUST be set.

O R flag: this bit MUST be cleared in the request.

O J flag: if this is the JOIN request, it should be set.

O TTL: Every overlay MAY specify a default TTL in terms of the size of the overlay peers. This default values MAY be obtained through the enrollment procedure. It is up to the source peer to decide the value of the TTL.

O Transaction ID: a 32 bit random number which is used to match the response with the request. The method to get the transaction ID is implementation dependent.

After the requests are generated, the forwarding operation described in section 8 MUST be followed to find out who is the next peer to the destination ID, and then send the requests directly to the next peer by using the established control connection between them.

Source peer should keep the state for the request and wait for the response. In order to make sure the reliability, the source peer MAY set a retransmit timer. If the timer fires, the source peer should send the request again until maximum retransmit times reached.

8.1.2. Processing the Response

When the peer receives a response from the network, the forwarding operation will decide how to process it. If the Source Peer ID in the response equals to the Peer ID of the forwarding peer, the forwarding peer knows it is the destination of the response and it will deliver the response to the upper layer and perform the message-specific processing.

Before the message-specific processing, the peer MUST check the message in the following rules:

1. Check the value in the TTL field whether it is zero. If it is, discard the response.
2. Make sure Destination Peer Info attribute and the Response Attribute are carried in the response. If any of them disappears, the peer MUST discard the response.
3. Extract Source Peer ID, destination ID, transaction ID and the message type from the message header, and match them with the pending requests. If there is no matched one, it means that the response MAY either arrive at a wrong destination or be a delayed response, therefore the response MUST be discarded.
8.2. Intermediate Peer Behavior

When the peer receives a message from the network, it first checks if it is the destination of the message. If it is not, it plays the intermediate peer role. For the intermediate peer, the main task is to forward the message through the overlay or relay the messages to their destination.

SEP introduces the hop-by-hop option which means that the intermediate peers should perform some message-specific actions in addition to forwarding operations. These message-specific actions would provide some information which will be carried in the message and learned by the source peer in the end. For example, source peer MAY want to get a route log of a request, so every intermediate peer should put itself into the request and later the route log will be returned to source peer.

SEP also sets a J flag in the JOIN request which means that the JOIN request has not been processed by any peer in this overlay. So when a peer receives a JOIN request, it would take the role of the bootstrap peer if J flag is set. What the bootstrap peer could do to the JOIN request is to clear the J flag and get the source transport address of the JOIN request, and then record it into the Source Reflexive Address attribute in the request.

Intermediate peer could decide whether the message is a request or response by only looking at the R flag in the message header.

8.2.1. Request Processing

When the intermediate peer receives a request, it first checks whether the hop-by-hop flag is set. If it is set, it should perform some message-specific actions according to the message type. Whether the H flag is set or not, the basic forwarding operations MUST be performed.

The intermediate peer should also check the J flag if the message is JOIN request. If the J flag is set, the intermediate peer should do some message-specific action. If it is not set, only basic forwarding operations is performed.

The TTL should be decreased by 1 and if the result is 0, the intermediate peer MUST discard the request.
8.2.2. Response Processing

When the intermediate peer receives a response, it should check the F flag and determine what kinds of routing mode will be preferred by the destination peer. If the F flag is set, it means the response should be relayed by the intermediate peer; if it is not, the response should be routed on the overlay level like what the requests are treated.

8.3. Destination Peer Behavior

8.3.1. Request Processing

When the peer receives a request, it should consult their routing states to decide whether it is the destination peer for the request. If it is the destination peer, some routine checks MUST be performed before the message-specific actions are to be performed.

1. Check whether TTL is zero. If it is, MUST discard the message;
2. Check whether the Source Peer Info attribute is in the request. If it is absent, MUST discard the message;

8.3.2. Response Generation and Sending the Response

After processing the request, the destination SHOULD send a response to the source peer. The rules below should be followed:

1. The Source Peer, Destination ID, message type and the transaction ID MUST not be changed in the response. These fields should be kept the same as the request.
2. Response Attribute MUST be included in the response;
3. Destination Peer Info MUST be included in the response.

How to send the response depends on the routing modes chosen by the destination peer. Please refer to the section 7.2.1 for more detail.

9. NAT Traversal

The existence of NAT makes the communication between peers becomes harder. IETF has developed a suite of tools, STUN/TURN/ICE to address the issue. SEP intends to reuse these tools and work with them in a harmonious way. For example, SEP provides a method to exchange the candidate for the ICE protocol and SEP has the capability for the peer to discover the STUN or TURN server.

An overlay is made up of the peers and the connections between peers. Peers could reach its neighbors directly. For some operations like
GET or PUT, source peers may only request the service provided by the destination peer and would not communicate with the destination peer any more. In that case, the requests are delivered over the established connections and later reach the destination peer. There is nothing needed to do to address the NAT traversal of the request. But for the response, if it is forwarded on the overlay layer, it also would not be interfered by the existence of the NAT for the same reason as the request; But if it is forwarded on the IP layer, NAT MAY fail the response. SEP introduces relaying peers to make response go back to the source peer by traversing NAT.

For some other operations like JOIN, SearchPeer, source peers not only request the service provided by the destination peer, but also expect the direct connection between them. If a new peer wants to join the overlay, it sends JOIN request to the destination peer. The destination peer for the request must be the new peer’s neighbor; therefore they would communicate directly later. SEP provides a method to exchange the candidates for the source peers and destination peers and ICE could make use of them to attempt to find out one or more candidate pairs by which they could reach each other directly even in the presence of NAT between them.

Before a peer joins the overlay, it should initiate an enrollment procedure. The joining peer MUST learn some bootstrap peers with public address. These public bootstrap peers could provide STUN service for other peers, therefore the joining peer could learn whether it is behind NAT or not with the help from these public bootstrap peer or some other STUN servers.

The peer MAY uses STUN protocol to detect whether it is behind the NAT. If it is, it may need a TURN server if it is behind a p2p-unfriendly NAT. Service peer discovery method proposed in SEP could be used to find service peers which could provide TURN service. Alternatively, it also could get a TURN server in other ways, for example, from the enrollment server.

9.1. Gather ICE candidates

A peer could learn whether it is behind NAT by means of STUN. If it is, it should gather ICE candidates and put them into the Source Peer Info or Destination Peer Info respectively. By using these two attributes, the source peer and the destination peer could exchange the ICE candidates with each other and the ICE process would be started as suggested by [ICE].

The method to gather ICE candidates is similar to that described in [ICE]. ICE candidates are often carried by using SDP, but the Peer Address Info is used in the SEP.
9.2. Response from the Destination Peer to the Source Peer

If the source peer is on the public Internet, the response could be sent back directly from the destination peer to the source peer. If the source peer is behind the NAT, it should either rely on the relaying peer to relay response or use the overlay native routing. It should follow the forwarding operations described in section 7.

9.2.1. How to Make JOIN Response Return to Source Peer

JOIN message has a little difference from other message, because the joining peer has not joined the overlay and has no neighbors at that time. What the joining peer knows are only one or more bootstrap peers which are required to be on the public Internet.

In order to make the JOIN response return to the source peer in the presence of NAT, the joining peer SHOULD set the J flag in the message header to show that the JOIN request has not been processed by any peer in the overlay. In the meantime, the joining peer SHOULD get the transport address of the bootstrap peer which will be the next hop of the JOIN request and fill it into the Relaying Peer Info. After finishing above tasks, the joining peer sends the JOIN request to the bootstrap peer.

The bootstrap peer SHOULD check the J flag and decide whether it is the bootstrap peer of the JOIN request. If it is the bootstrap peer, it should record the source transport address of the JOIN request and fill it into the Source Reflexive Address attribute. The J flag MUST be cleared by the bootstrap peer.

When the destination peer receives the JOIN request, if the source peer is behind NAT, sending the response by routing through the overlay won’t make sense in that the joining peer has not been known by any peer in the overlay. So the only choice in SEP is for destination peer to relay the response to the source peer with the help of the relaying peer. In that case, the destination peer will send the response directly to the relaying peer and SHOULD put the Source Reflexive Address attribute unmodified into the response.

After the bootstrap peer (now, it plays relaying peer role) receives JOIN response, it checks whether Source Reflexive Address attribute is carried in the response. If it is, JOIN response will be forwarded to the transport address which could be got from the Source Reflexive Address. Then the JOIN response returns to the source peer.
9.3. Exchange Candidates for the Direct Communication

Some operations may require a direct connection after the associated transaction finishes. But due to the existence of NAT, both peers should exchange their own candidates with each other and then use ICE to find out one or more workable candidate pairs. The transaction before the direct connection could be used as a signaling channel to exchange candidates. In order to achieve this, the source peer SHOULD set the D flag in the message header if it requires a later direct connection with the destination peer.

After exchange candidates, ICE process will be started. (TO DO: how is SEP seamlessly integrated with the ICE process will explored in the next version.)

10. Service Discovery

As pointed by the descriptions in section 2.1 of [concept], some additional services other than routing and storage service will be needed. For example, STUN or STUN relay service may be required to allow the overlay to form and operate in the presence of NAT. So the distributed database on the top of the peer protocol should store some information about these services. For example, it may need to store information about which peers offer which services.

SEP provides a message LookUpServicePeer for the peer to attempt to collect the service peers providing a specific service. The method proposed in the SEP is to let the peer in need of a specific service talks to as much peers as possible and ask them to search service peers for it while the packet is routed through the overlay. If LookUpServicePeer messages will be sent several times, we could choose different destination IDs to make the message go through different paths, so that the source peer will have chance to ask much more peers to search peers by looking up its local states, for example, routing states which records some information about service peer.

What information SEP intends to get is to learn the reachability information of these service peers. It does not care about whether the service peers is capable of serving the peers in need of this service. With several service peers in mind, the peer in need of the service could try all service peers until they get served. As for how to choose the best service peer, it is hard to achieve. But in SEP, if every peer which provides a specific service could update its service status and keep it in the routing states as described in section 2.1, the peers who search service peers may choose by their associated service status.
11. Security Considerations

Security considerations will be discussed in the next version.

12. IANA Considerations

<table>
<thead>
<tr>
<th>Message</th>
<th>Message Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOIN</td>
<td>0</td>
</tr>
<tr>
<td>LEAVE</td>
<td>1</td>
</tr>
<tr>
<td>KEEPALIVE</td>
<td>2</td>
</tr>
<tr>
<td>NOTIFY</td>
<td>3</td>
</tr>
<tr>
<td>UPDATE</td>
<td>4</td>
</tr>
<tr>
<td>SearchPeer</td>
<td>5</td>
</tr>
<tr>
<td>TRANSFER</td>
<td>6</td>
</tr>
<tr>
<td>PUT</td>
<td>7</td>
</tr>
<tr>
<td>GET</td>
<td>8</td>
</tr>
<tr>
<td>REMOVE</td>
<td>9</td>
</tr>
<tr>
<td>LookUpServicePeer</td>
<td>10</td>
</tr>
</tbody>
</table>

13. Acknowledgements

A team of guys contribute to this document. Thanks for the effort from Alex, Watw, Justin and Rake.

14. References

14.1. Normative References


[ Concept ] Bryan, D., Matthews, P., Shim, E., Willis, D., "Concepts and Terminology for Peer to Peer SIP " Internet Draft
14.2.  Informative References

[STUN Discovery] XingFeng Jiang, "A mechanism to discover STUN/TURN nodes in P2PSIP" Internet Draft draft-jiang-p2psip-STUN-discovery (Work in Progress)

Authors’ Addresses

XingFeng Jiang
Huawei Tech.
Huihong Mansion,No.91 Baixia Rd.
Nanjing, Jiangsu 210001
P.R.China
Phone: +86(25)84565468
Email: jiang.x.f@huawei.com

HeWen Zheng
Huawei Tech.
Huihong Mansion,No.91 Baixia Rd.
Nanjing, Jiangsu P.R.China
Phone: +86(25)84565467
Email: hwzheng@huawei.com