

P2PSIP  
Internet-Draft  
Intended status: Standards Track  
Expires: June 2009  
December 16, 2008

H. Song  
X. Jiang  
Huawei

**Diagnose P2PSIP Overlay Network**  
**draft-zheng-p2psip-diagnose-04.txt**

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 June 16, 2009.

Abstract

This document describes P2PSIP diagnostics. It describes the usage scenarios and defines several simple methods for the diagnostics in P2PSIP overlay network. It also describes types of diagnostic information which are useful for the connection and node status monitoring. An optional diagnostic server is also defined for the overview of the overall health of the overlay. The methods and message formats are specified as extensions to P2PSIP base protocol RELOAD.



## Table of Contents

<a href="#">1.</a>	<a href="#">Introduction.....</a>	<a href="#">3</a>
<a href="#">1.1.</a>	<a href="#">Usage Scenarios.....</a>	<a href="#">3</a>
<a href="#">2.</a>	<a href="#">Terminology.....</a>	<a href="#">4</a>
<a href="#">3.</a>	<a href="#">Motivation.....</a>	<a href="#">4</a>
<a href="#">4.</a>	<a href="#">Overview of Functions.....</a>	<a href="#">5</a>
<a href="#">5.</a>	<a href="#">Packets Formats.....</a>	<a href="#">7</a>
<a href="#">5.1.</a>	<a href="#">Message Codes.....</a>	<a href="#">7</a>
<a href="#">5.2.</a>	<a href="#">Message payloads.....</a>	<a href="#">8</a>
<a href="#">5.2.1.</a>	<a href="#">Error Codes and Sub-Codes.....</a>	<a href="#">8</a>
<a href="#">5.2.2.</a>	<a href="#">diagnostics information.....</a>	<a href="#">9</a>
<a href="#">6.</a>	<a href="#">Message.....</a>	<a href="#">10</a>
<a href="#">6.1.</a>	<a href="#">Inspect.....</a>	<a href="#">10</a>
<a href="#">6.2.</a>	<a href="#">Path_Track.....</a>	<a href="#">12</a>
<a href="#">6.3.</a>	<a href="#">Echo.....</a>	<a href="#">15</a>
<a href="#">6.4.</a>	<a href="#">Error responses.....</a>	<a href="#">17</a>
<a href="#">7.</a>	<a href="#">Diagnostic Server.....</a>	<a href="#">18</a>
<a href="#">8.</a>	<a href="#">Security Considerations.....</a>	<a href="#">18</a>
<a href="#">9.</a>	<a href="#">IANA Considerations.....</a>	<a href="#">18</a>
<a href="#">10.</a>	<a href="#">Acknowledgments.....</a>	<a href="#">19</a>
<a href="#">11.</a>	<a href="#">References.....</a>	<a href="#">20</a>
<a href="#">11.1.</a>	<a href="#">Normative References.....</a>	<a href="#">20</a>
<a href="#">11.2.</a>	<a href="#">Informative References.....</a>	<a href="#">21</a>
	<a href="#">Author's Addresses.....</a>	<a href="#">22</a>

## 1. Introduction

P2P systems are self-organizing and ideally require no network management in the traditional sense to set up and to configure individual P2P nodes. P2P service providers may however contemplate usage scenarios where some monitoring and diagnostics are required. We present a simple connectivity test and some useful diagnostic information that may be used in such diagnostics.

### 1.1. Usage Scenarios

The common usage scenarios for P2P diagnostics can be broadly categorized in three classes:

a. Automatic diagnostics built into the P2P overlay routing protocol. Nodes perform periodic checks of known neighbors and remove those nodes from the routing tables that fail to respond to connectivity checks [Handling Churn in a DHT]. The unresponsive nodes may however be only temporarily disabled due to some local cryptographic processing overload, disk processing overload or link overload. It is therefore useful to repeat the connectivity checks to see if such nodes have recovered and can be again placed in the routing tables. This process is known as 'failed node recovery' and it can be optimized as described in the reference [Handling Churn in a DHT].

b. P2P system diagnostics to check the overall health of the P2P overlay network, the consumption of network bandwidth, problem links and also checks for abusive or malicious nodes. This is not a trivial problem and has been studied in detail for content and streaming P2P overlays, such as for example in [Diagnostic Framework].

Similar work has been reported more recently for P2PSIP overlays as applied to the P2PP protocol [Diagnostics and NAT traversal in P2PP].

c. Diagnostics for a particular node to follow up an individual user complaint. In this case a technical support person may use a desktop sharing application with the permission of the user to determine remotely the health and possible problems with the malfunctioning node. Part of the remote diagnostics may consist of simple connectivity tests with other nodes in the P2PSIP overlay. The simple connectivity tests are not dependent on the type of P2PSIP overlay and they are the topic of this memo. Note however that other tests may be required as well, such as checking the health and performance of the user's computer or mobile device and also checking the link bandwidth connecting the user to the Internet.



## 2. Terminology

The concepts used in this document are compatible with "Concepts and Terminology for Peer to Peer SIP" [[I-D.ietf-p2psip-concepts](#)] and the P2PSIP base protocol RELOAD[I-D.ietf-p2psip-base].

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].

## 3. Motivation

In the last few years, overlay networks have rapidly evolved and emerged as a promising platform to deploy new applications and services in the Internet. One of the reasons overlay networks are seen as an excellent platform for large scale distributed systems is their resilience in the presence of failures. This resilience has three aspects: data replication, routing recovery, and static resilience. Routing recovery algorithms are used to repopulate the routing table with live nodes when failures are detected. Static resilience measures the extent to which an overlay can route around failures even before the recovery algorithm repairs the routing table. Both routing recovery and static resilience relies on accurate and timely detection of failures.

As described in "Security requirements in P2PSIP" [I-D.matuszewski-p2psip-security-requirement], there are some malfunctioning or badly behaving peers in P2PSIP overlay, those peers may be disabled peers, congested peers or peers behaving with misrouting, and the impact of those peers in the overlay network is degradation of quality of service provided collectively by the peers in the overlay network or interruption of those services. It is desirable to identify malfunctioning or badly behaving peers through some diagnostics tools, and exclude or reject them from the P2PSIP system. Besides those faults, node failures may be caused by underlying failures, for example, when the IP layer routing failover speed after link failures is very slow, then the recovery from the incorrect overlay topology may also be slow. Moreover, if a backbone link fails and the failover is slow, the network may be partitioned, which may lead to partitions of overlay topologies and inconsistent routing results between different partitioned components.

Some keep-alive algorithms based on periodically probe and acknowledge enable accurate and timely detection of failures of one peer's neighbors [[Overlay-Failure-Detection](#)], but those algorithms only can detect the disabled neighbors using the periodical method,



it may not be enough for operating the overlay network by service providers.

One general P2PSIP overlay diagnostics protocol supporting periodical method and on-demand method for node failures and network failures is desirable. This document describes one general P2PSIP overlay diagnostics protocol useful for P2PSIP base protocol and it is a good match for some keep-alive algorithms in the P2P or P2PSIP overlay itself.

#### 4. Overview of Functions

As a diagnostics protocol, P2PSIP diagnostics protocol is mainly used to detect and localize failures or monitor performance in P2PSIP overlay network. It provides mechanisms to detect and localize malfunctioning or badly behaving peers including disabled peers, congested peers and misrouting peers. It provides a mechanism to detect direct connectivity or connectivity to the specified peer, a mechanism to detect availabilities of specified resource records and a mechanism to discover P2PSIP overlay topology and the underlay topology failures.

The P2PSIP diagnostics protocol defines Inspect and Path\_Track methods for connection quality check and retrieval of diagnostic information, as well as Echo method for efficient diagnostics in the administrative overlay and the Error response to these methods. Essentially it reuses P2PSIP base protocol specification and then introduces the new diagnostics methods. P2PSIP diagnostics protocol strictly follows the P2PSIP base protocol specification on the messages routing, transporting and NAT traversal etc. The diagnostic methods are however P2PSIP protocol independent.

In this document, we mainly describe how to detect and localize those failures including disabled peers, congested peers, misrouting behaviors and underlying network faults in P2PSIP overlay network through a simple and efficient mechanism. This mechanism is modeled after the ping/traceroute paradigm: ping (ICMP echo request [[RFC792](#)]) is used for connectivity checks, and traceroute is used for hop-by-hop fault localization as well as path tracing. This document specifies a "ping" mode (by defining the Inspect method) and a "traceroute" mode (implemented differently with trusted overlays such





as operator deployed P2PSIP overlays, compared with untrusted overlays)for diagnosing P2PSIP overlay network.

An Inspect request message is forwarded by the intermediate peers along the path and then terminated by the responsible peer, and after optional local diagnostics, the responsible peer returns an Inspect response message. If an error is found when routing, an Error response is sent to the initiator node by the intermediate peer.

In "Traceroute" mode, we classify the diagnostics into two application scenarios.

(1) In trusted p2p overlays, we use an Echo request message, it is received and disposed by each peer along the routing path, and each peer along the path returns an Echo response message with optional local diagnostics information including the result and causes if existing.

(2) In untrusted p2p overlays, we define a simple Path\_Track method for retrieving diagnostics information iteratively. First, the initiating node asks its neighbor A which is the next hop node to the destination ID, and then retrieve the next hop node B information, along with optional diagnostic information of A, to the initiator node. Then the initiator node asks the next hop node B(directly or symmetric routing) to get the further next hop node C information and diagnostic information of B. This step can be iterative until the request reaches responsible node D for the destination ID, and retrieve diagnostic information of node D, or terminates by some failures that prevent the process.

One approach these tools can be used is to detect the connectivity to the specified peer or the availability of the specified resource-record through P2PSIP Inspect operation once the overlay network receives some alarms about overlay service degradation or interruption, if the Inspect fails, one can then send a P2PSIP Traceroute(iterative Path\_Track or Echo) to determine where the fault lies.

The diagnostic information MUST be only provided to authorized peers. Some diagnostic information can be authorized to all the participants in the P2PSIP overlay, and some other diagnostic information can only be provided to the authorization peer list of each diagnostic information according to the local or overlay policy. The



authorization mainly depends on the kinds of the diagnostic information and the administrative considerations.

## 5. Packets Formats

This document extends the P2PSIP base protocol to carry diagnostics information. Considering special usage of diagnostics, this document defines three simple methods Inspect, Path\_Track and Echo, as well as related Error codes and some useful diagnostics information.

As described in the P2PSIP base protocol, each message has three parts. This specification is consistent with the format.

```
+-----+
|   Forwarding Header   |
+-----+
|   Message Contents   |
+-----+
|         Signature     |
+-----+
```

### 5.1. Message Codes

The mechanism defined in this document follows P2PSIP base protocol specification, the new request and response message use the message format specified in P2PSIP base protocol messages. Different types of messages convey different message contents following the forwarding header according to the protocol design. Please refer to P2PSIP base protocol [[I-D.ietf-p2psip-base](#)] for the detailed format of forwarding header.

This document reuses the Error response in the base protocol and defines new Error codes to carry different failure reports to the initiator node when failure is detected during diagnostics.

Name	Message Code
Error	0xFFFF

This document introduces three types of message:

Name	Message Code
Inspect request	29
Inspect response	30
Path_Track request	31
Path_Track response	32
Echo request	33
Echo response	34

The final message code will be assigned by IANA as specified in section 14.4 of [[I-D.ietf-p2psip-base](#)].

## 5.2. Message payloads

As an extension to P2PSIP base protocol, a P2PSIP diagnostics protocol message content contains one message code following by its payloads. Please refer to P2PSIP base protocol [[I-D.ietf-p2psip-base](#)] for the detailed format of Message Contents.

In addition to the newly introduced methods, this document extends the Error codes defined in P2PSIP base protocol specification.

### 5.2.1. Error Codes and Sub-Codes

This document extends the Error response method defined in the P2PSIP base protocol specification to describe the result of diagnostics.

This document introduces new Error Codes as below:

Code Value	Error Code Name
8	Underlay Destination Unreachable
9	Underlay Time exceeded
10	Message Expired
11	Upstream Misrouting
12	Loop detected
13	TTL hops exceeded

The final error codes will be assigned by IANA as specified in [section 14.5](#) of the p2psip base protocol [[I-D.ietf-p2psip-base](#)].

This document introduces several types of error information in the error\_info field for Error Code 8 as an example:



error\_info:

- net unreachable
- host unreachable
- protocol unreachable
- port unreachable
- fragmentation needed
- source route failed

Editor note: We may need more discussion here to see if we need to define an additional sub-code field for the error information. Sub-code is easier for the machine to process while various text is comfortable for a man to read.

### **5.2.2. diagnostics information**

This document introduces some new diagnostics information conveyed in the message payload, including: the number of hops that the message traverses, the underlay TTL specified, the timestamp of initiating the request message, the timestamp of receiving the request message, and the expiration time of the request message, the processing power, the bandwidth, the number of entries in one's neighbor table, etc. They are defined as below. Additional diagnostic information have been proposed in [section 9](#) of the p2psip base protocol draft [I-D.ietf-p2psip-base].

HopCounter (8 bits): This byte only appears in diagnostic responses. It must be exactly copied from the TTL field of the forwarding header in the received request. Then this information is sent back to the request initiator to compute the hops that the message traverses in the overlay.

UnderlayTTL (8 bits): It indicates the underlay TTL which the intermediate peer must adopt when forwarding the diagnostic requests, it is specified by the initiator. If the value is 0, then the intermediate peer must ignore this field, and use the underlay TTL with its local configuration.

TimestampInitiated (64 bits): The time-of-day (in seconds and microseconds, according to the sender's clock) in NTP timestamp format [[RFC4330](#)] when the P2PSIP Overlay diagnostic request is sent. It can be carried in the diagnostic response message from the receiver; certainly it first appears in the diagnostic request message;





TimestampReceived (64 bits): it is in a diagnostic response message as the time-of-day (according to the receiver's clock) in NTP timestamp format [[RFC4330](#)] that corresponds to the time that the P2PSIP Overlay diagnostic request was received;

Expiration(64 bits): the expiration time of the request message, it is the time-of-day in NTP timestamp format [[RFC4330](#)]. It can be used to mitigate the replay attack to the destination peer and overlay network.

ProcessPower (32 bits): A single value element containing an unsigned 32-bit integer specifying the processing power of the node in unit of MIPS.

Bandwidth (32 bits): A single value element containing an unsigned 32-bit integer specifying the bandwidth of the node in unit of Kbps.

Routing\_Table\_Size (32 bits): A single value element containing an unsigned 32-bit integer representing the number of peers in the peer's routing table. The administrator of the overlay may be interested in statistics of this value for the consideration such as routing efficiency.

StatusInfo (8 bits): A single value element containing an unsigned byte representing whether or not the node is in congestion status.

## 6. Message

All P2PSIP base protocol requests and responses use the common forwarding header followed by the message contents.

This document defines Inspect and Path\_Track methods, and introduces the new Echo message to detect and localize failures in P2PSIP overlay network. The Error Codes to these requests are defined in [section 5.2.1](#) of this spec.

### **[6.1. Inspect](#)**

In P2PSIP base protocol, Ping is used to test connectivity along a path. However, connectivity quality can not be measured well without some useful information, such as the timestamp and HopCounter. Here we define a new method Inspect for connectivity quality check purposes. See below for the Inspect formats.



Inspect Request:

```
struct {  
    uint8  UnderlayTTL;  
    uint64 TimestampInitiated;  
    uint64 Expiration;  
} InspectReq;
```

Inspect Response:

```
struct {  
    uint8  HopCounter;  
    uint64 TimestampReceived;  
    uint64 Expiration;  
} InspectAns;
```

Any intermediate node which receives the Inspect request must adopt the UnderlayTTL for the next hop forwarding. If the value equals to 0, the intermediate peer must ignore this value and determine the underlay TTL using local configuration. The requirement here is that one may want to limit each hop underlay TTL from the initiator to the destination, if the underlay TTL expires somewhere, one may think that the link there is not of good quality.

Each intermediate peer receiving the Inspect request/response should check the Expiration value (NTP format) to determine if the message expired. If the message expired, the intermediate peer should generate a "Message Expired" Error to the initiator node, and discard the message. The responsible node for the request or response MUST check this value to determine if this message should be ignored.

The responsible peer of the Inspect request must exactly copy the TTL field value in the forwarding header to the HopCounter value in the response, and meanwhile, it should generate a NTP format timestamp to indicate the received time.

The initiator node, as well as the responding peer, can compute the overlay One-Way-Delay time through the value in TimestampReceived and the TimestampInitiated field.

Editor note: We need more discussion here because time synchronization is a barrier in open Internet environment, while in the operator's network, it is not so tricky.

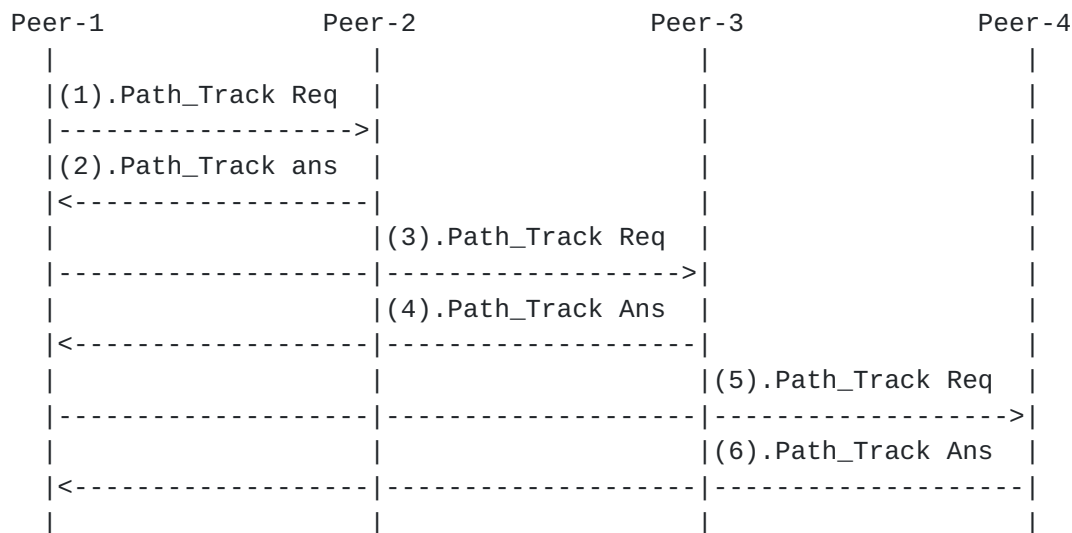
The initiator node receiving the Inspect response must check the HopCounter field and compute the overlay hops to the destination peer for the statistics of connectivity quality from the perspective of overlay hops.



Inspect is also used to detect possible failures in the specified path of P2PSIP overlay network. If disabled peers, misrouting behavior and underlying network faults are detected during the routing process, the Error responses with Error codes and descriptions, must be sent to the initiator node immediately. See [section 6.4](#) for the details.

## 6.2. Path\_Track

We define a simple Path\_Track method to retrieve the diagnostic information from the intermediate peers along the routing path. At each step of the Path\_Track request, the responsible peer responds to the initiator node with the status information of itself whether or not congested, its processing power, its available bandwidth, the number of entries in its neighbor table, its uptime, his identity and network address information, and the next hop peer information.



Path\_Track example

A Path\_Track request must specify which diagnostic information is requested by setting different bits in the flag contained in the Path\_Track request payload. If the flag is clear, then the Path\_Track request is only used for asking the next hop information, in this case the iterative mode of Path\_Track is used only for checking the liveness of the peers along the routing path. The Path\_Track request can be routed directly or through the overlay based on the local policy of the initiator node.



```
Path_Track request:
    struct {
        Destination          destination;
        Uint32                dFlags;
        uint64                Expiration;
    } RathTrackReq;
```

destination : The destination which the requester is interested in. This may be any valid destination object, including a Node-ID, compressed ids, or Resource-ID.

dFlags : An unsigned 32-bit integer indicating which kind of diagnostic information the initiator interested in. The initiator sets different bits to retrieve different kinds of diagnostic information. If dFlags is clear, then no diagnostic information is conveyed in the Path\_Track response. If dFlag is set to 0xFFFFFFFF, then all diagnostic information kinds are requested. The kinds of diagnostic information including: status information, its processing power, its available bandwidth, the number of entries in its neighbor table, its uptime, etc. The mapping between the bits in the dFlags and the diagnostic information kind is as below:

StatusInfo Flag(0x0001) : if set, the status information of the responding peer is requested;

Routing\_Table\_Size Flag(0x0002) : if set, the number of entries in the responding peer's neighbor is requested;

ProcessPower Flag(0x0004) : if set, the processing power information of the responding peer is requested;

Bandwidth Flag(0x0008) : if set, the bandwidth information of the responding peer is requested;

[TODO: The bits in the dFlags should also map to the diagnostic information defined in [section 9](#) of p2psip base draft.]

A response to a successful PathTrackReq is a PathTrackAns message. There is a general diagnostic information part based on the flags.

Path\_Track response:

```
struct {
    Destination      next_hop;
    Diagnostic_Info  diag_info_list<0..2^5-1>;
    uint64           Expiration;
} RathTrackAns;
```

next\_hop : The information of the next hop node from the responding intermediate peer to the destination node. If the responding peer is the responsible peer for the destination ID, then the next\_hop node ID equals the responding node ID, and after that the initiator must stop the iterative process.

diag\_info\_list: The diagnostic information from the responding peer.

The TLV structure for Diagnostic\_Info is as the following:

```
struct {
    uint8      Kind-ID;
    uint8      length;
    Opaque     diagnosic_information<0..2^8-1>;
} Diagnostic_Info;
```

Various kinds of diagnostic information can be retrieved, some of them are defined in this document, and some of them are defined in the p2psip base draft. This document introduces additional three new data kind-IDs as below:

Kind	Kind-ID
StatusInfo	16
ProcessPower	17
Bandwidth	18

The final kind-IDs will be assigned by IANA as specified in [section 14.2](#) of the p2psip base protocol [[I-D.ietf-p2psip-base](#)].

As for the Path\_Track responses, whether or not sending back certain kind of diagnostic information to the initiator node depends on (1) the dFlags (2)the authorization policy.

Failures may be detected during the process, after that an Error response should be reported to the initiator node immediately.





### **6.3. Echo**

An Echo request message is used to retrieve the diagnostic information of the specified path in administrative p2p overlays where all the peers in the overlay are trusted or based on specific authorization. For example, it can be used in a p2p overlay where all peers deployed by the operator to provide services to the customers(clients), where the diagnostics happens between peers in the p2p overlay. For the untrusted p2p overlays, the Echo method must be used with care for the consideration of potential DoS attack.

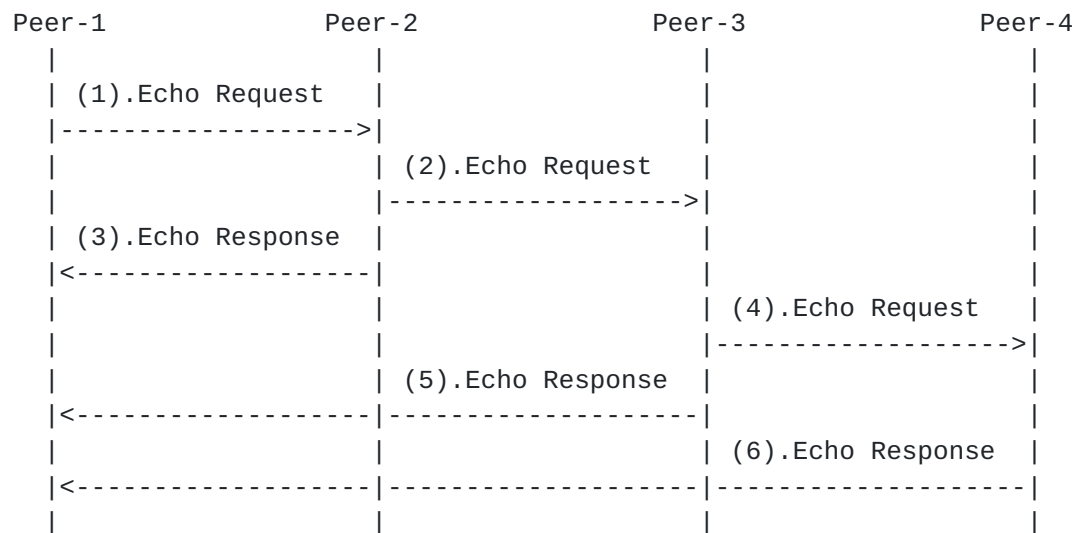
An Echo request is normal P2PSIP base protocol message; it can be initiated by any node in the administrative p2p overlay which supports P2PSIP base protocol specification.

An Echo request must specify which diagnostic information it is interested in by setting different bits in the dFlags contained in the request payload. If all diagnostic flags are clear, the response is a simple Echo response containing no additional diagnostic information.

Any intermediate peer along the Echo request path should forward the Echo request to the next hop, and then returns an Echo response to the initiator node, along with the diagnostic information indicated by the dFlags in the Echo request.

As for the Echo responses, whether or not sending back certain kind of diagnostic information to the initiator node depends on (1) the dFlags (2)the authorization policy.

Any Echo request/response whose time in the Expiration field expired should be ignored.



P2PSIP Echo example

Echo request:

```

Struct {
    Uint32 dFlags;
    uint64 Expiration;
}EchoReq
  
```

dFlags (32 bits): An unsigned 16-bit integer indicating which kind of diagnostic information the initiator interested in. The initiator sets different bits to retrieve different kinds of diagnostic information. If dFlags is clear, then no diagnostic information is conveyed in the Echo response. See [section 6.2](#) for the mapping between the bits in the dFlags and the diagnostic information kinds.

Expiration: See [section 5.2.2](#) for the meaning.

Echo response:

```

Struct {
    Diagnostic_Info diag_info_list<0..2^5-1>;
    uint64 Expiration;
}EchoAns
  
```

The peer may find misrouting behaviors or the underlay failures during the Echo process, then an Error response should be generated and send back to the initiator node. See [section 6.4](#) for details.



#### **6.4. Error responses**

In p2psip overlay, the error response can be generated by the intermediate peer or responsible peer, to a diagnostic message or other messages. All error responses contain the Error code followed by the subcode and descriptions if existed.

When a request arrives at a peer, if the peer's responsible ID space does not cover the destination ID of the request, then the peer continues to forward this request according to the overlay specified routing mode.

When a request arrives at a peer, the peer may find some connectivity failures or malfunction peers through the analysis of via list or underlay error messages. The peer should report the error responses to the initiator node. The malfunction node information should also be reported to the initiator node in the error message payload.

The peer should return an Error response with the Error Code 8 "Underlay Destination Unreachable" when it receives an ICMP message with "Destination Unreachable" information after forwarding the received request.

The peer should return an Error response with the Error Code 9 "Underlay Time Exceeded" when it receives an ICMP message with "Time Exceeded" information after forwarding the received request.

The peer should return an Error response with the Error Code 10 "Message Expired" when it finds that the message expires the time field Expiration contained in the message payload.

The peer should return an Error response with Error Code 11 "Upstream Misrouting" when it finds its upstream peer disobeys the routing rules defined in the overlay. The immediate upstream peer information should also be conveyed to the initiator node.

The peer should return an Error response with Error Code 12 "Loop detected" when it finds a loop through the analysis of via list.

The peer should return an Error response with Error Code 13 "TTL hops exceeded" when it finds that the TTL field value is no more than 0 when forwarding.



## 7. Diagnostic Server

For analyzing the overall health of the overlay network, a diagnostic server can be used. The diagnostic server will collect diagnostic information gathered, for example using the protocol specified in this document, from the overlay nodes.

Administrator of the overlay can get a clear knowledge of the overlay and take certain strategy when the overall performance of the overlay changes(e.g. change the ratio of number between peers and clients when most peers are in "busy" status).

Through the statistic information, the diagnostic server also knows whether the overall storage or processing power is enough or not for the overlay requirement.

It can also affirm the malfunction node through the statistics of diagnostic information and degrade that peer to be a client or eject it from the overlay.

A protocol between the diagnostic server and the overlay nodes is needed.

## 8. Security Considerations

The Echo method may potentially cause DoS attack to the initiator, though this implementation is more efficient than using iterative mode of Path\_Track operation.

An advice is to use the efficient Echo operation in administrated P2PSIP overlay and use the pacing-style Path\_Track operation in the untrustworthy P2PSIP overlay network, certainly, the probability of this type of DoS attack is very low because the overlay is distributed and then it is very hard for the attacker to know the accurate Peer-IDs and attack most of all peers simultaneously.

## 9. IANA Considerations

Message Code: this document introduces three new type of message to the "RELOAD Message Code" Registry as below:

Message Code Name	Code Value	RFC
inspect_req	29	RFC-AAAA
inspect_ans	30	RFC-AAAA
path_track_req	31	RFC-AAAA
path_track_ans	32	RFC-AAAA
echo_req	33	RFC-AAAA
echo_ans	34	RFC-AAAA

Error Code: this document introduces some new Error Codes to the "RELOAD Message Code" Registry as below:

Code Value	Error Code Name
8	Underlay Destination Unreachable
9	Underlay Time exceeded
10	Message Expired
11	Upstream Misrouting
12	Loop detected
13	TTL hops exceeded

Data Kind-ID: This document introduces additional data kind-IDs to the "RELOAD Data Kind-ID" Registry as below:

Kind	Kind-ID
StatusInfo	16
ProcessPower	17
Bandwidth	18

## 10. Acknowledgments

We would like to thank Zheng Hewen for the contribution of the initial version of this draft. We would also like to thank Bruce Lowekamp, Salman Baset, Henning Schulzrinne, Roni Even and Jiang Haifeng for the email discussion and their valued comments, and thank Henry Sinnreich for contributing to the usage scenarios in the Introduction.





## 11. References

### **11.1. Normative References**

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [2] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", [RFC 2234](#), Internet Mail Consortium and Demon Internet Ltd., November 1997.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2234] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", [RFC 2234](#), Internet Mail Consortium and Demon Internet Ltd., November 1997.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", [RFC 3261](#), June 2002.
- [RFC792] Postel, J., "Internet Control Message Protocol", STD5, [RFC 792](#), September 1981.
- [RFC4330] Mills, D., "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI", [RFC 4330](#), January 2006.
- [RFC4981] J. Risson, "Survey of Research towards Robust Peer-to-Peer Networks: Search Methods", [RFC 4981](#), September 2007.
- [I-D.ietf-p2psip-concepts] Bryan, D., "Concepts and Terminology for Peer to Peer SIP", [draft-ietf-p2psip-concepts-00](#) (work in progress), June 2007.
- [I-D.ietf-p2psip-base] Jennings, C., "REsource LOcation And Discovery (RELOAD) Base Protocol", [draft-ietf-p2psip-base-01](#) (work in progress), December 2008.
- [I-D.jiang-p2psip-sep] X. Jiang, "Service Extensible P2P Peer Protocol", [draft-jiang-p2psip-sep-00](#) (work in progress), November 2007.
- [I-D.bryan-p2psip-requirement] D. Bryan, "P2PSIP Protocol Framework and Requirements", [draft-bryan-p2psip-requirements-00](#) (work in progress), July 2007



- [Overlay-Failure-Detection] S. Zhuang, "On failure detection algorithms in overlay networks", Proc. IEEE Infocomm, Mar 13-17 2005.
- [P2PSIP-Concepts-Terminology] Dean Willis, "P2PSIP Concepts and Terminology", <http://www3.ietf.org/proceedings/07jul/slides/p2psip-13.pdf>, July 2007
- [Handling Churn in a DHT] S. Rhea et al: "Handling Churn in a DHT". USENIX Annual Conference, June 2004
- [Diagnostic Framework] X. Jin et al: "A Diagnostic Framework for Peer-to-Peer Streaming", Hong Kong University and Microsoft, 2005
- [Diagnostics and NAT traversal in P2PP] G. Gupta et al: "Diagnostics and NAT Traversal in P2PP - Design and Implementation." Columbia University Report. June 2008

## **11.2. Informative References**

- [I-D.ietf-behave-rfc3489bis] Rosenberg, J., Huitema, C., Mahy, R., and D. Wing, "Simple Traversal Underneath Network Address Translators (NAT) (STUN)", [draft-ietf-behave-rfc3489bis-08](#) (work in progress), July 2007.
- [I-D.ietf-behave-turn] Rosenberg, J., Mahy, R., and C. Huitema, "Obtaining Relay Addresses from Simple Traversal Underneath NAT (STUN)", [draft-ietf-behave-turn-04](#) (work in progress), July 2007.
- [I-D.ietf-mmusic-ice] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Methodology for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", [draft-ietf-mmusic-ice-17](#) (work in progress), July 2007
- [I-D.bryan-p2psip-dsip] Bryan, D., "dSIP: A P2P Approach to SIP Registration and Resource Location", [draft-bryan-p2psip-dsip-00](#) (work in progress), February 2007.
- [I-D.baset-p2psip-p2pp] S. Baset, "Peer-to-Peer Protocol (P2PP)", [draft-baset-p2psip-p2pp-00](#) (work in progress), July 2007.



[I-D.Jennings-p2psip-asp] C. Jennings, "Address Settlement by Peer to Peer", [draft-jennings-p2psip-asp-00](#) (work in progress), July 2007.

[I-D.marocco-p2psip-xpp-pcan] Marocco, E. and E. Ivov, "XPP Extensions for Implementing a Passive P2PSIP Overlay Network based on the CAN Distributed Hash Table", [draft-marocco-p2psip-xpp-pcan-00](#) (work in progress), June 2007.

[I-D.matthews-p2psip-hip-hop] Cooper, E., "A Distributed Transport Function in P2PSIP using HIP for Multi-Hop Overlay Routing", [draft-matthews-p2psip-hip-hop-00](#) (work in progress), June 2007.

[I-D.matuszewski-p2psip-security-requirement] M. Matuszewski, "Security requirements in P2PSIP", [draft-matuszewski-p2psip-security-requirements-04](#) (work in progress), November 2008

#### Author's Addresses

Haibin Song  
Huawei  
10F, Huihong Mansion, No. 91 Baixia Rd., Nanjing, China  
Phone: +86-25-84565867  
Email: melodysong@huawei.com

Xingfeng Jiang  
Huawei  
10F, Huihong Mansion, No. 91 Baixia Rd., Nanjing, China  
Phone: +86-25-84565868  
Email: jiang.x.f@huawei.com

#### Full Copyright Statement

Copyright (C) The IETF Trust (2008).

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 or 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](mailto:ietf-ipr@ietf.org).