P2PSIP Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 12, 2010

H. Song
X. Jiang
R. Even
Huawei
D. Bryan
Cogent Force, LLC
December 09, 2009

P2PSIP Overlay Diagnostics
draft-ietf-p2psip-diagnostics-02

Abstract

This document describes mechanisms for P2PSIP diagnostics. It describes the usage scenarios and defines several simple methods for performing diagnostics in P2PSIP overlay networks. It also describes the diagnostic information which is useful for the connection and node status monitoring. The methods and message formats are specified as extensions to P2PSIP base protocol RELOAD.

Table of Contents

1. Introduction

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-requirements], there are a number of situations in which some peers in a P2PSIP overlay may malfunction or behave badly. For example, these peers may be disabled peers, congested peers or peers misrouting messages, and the impact of those peers on the overlay network may be a degradation of quality of service provided collectively by the peers in the overlay network or an interruption of those services. It is desirable to identify malfunctioning or badly behaving peers through diagnostic tools, and exclude or reject them from the P2PSIP system. Besides those faults, node failures may be caused by underlying failures, for example, the recovery from an incorrect overlay topology may be slow when the IP layer routing failover speed after link failures is very slow. Moreover, if a backbone link fails and the failover is slow, the network may be partitioned, leading to partitions of overlay topologies and inconsistent routing results between different partitioned components.

Some keep-alive algorithms based on periodic probe and acknowledge mechanisms enable accurate and timely detection of failures of one peer's neighbors [Overlay-Failure-Detection], but these algorithms by themselves can only detect the disabled neighbors using the periodic method, it may not be enough for service providers operating the overlay network.

A single, general P2PSIP overlay diagnostic framework supporting periodic and on-demand methods for detecting node failures and network failures is desirable. This document describes a general P2PSIP overlay diagnostic extension to the P2PSIP base protocol and it is a good compliment to keep-alive algorithms in the P2P or P2PSIP overlay itself.

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 [RFC2119].

3. Diagnostic Scenarios

P2P systems are self-organizing and ideally require no network management in the traditional sense to set up and to configure individual P2P nodes. However, P2P service providers may 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.

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]. However, the unresponsive nodes may only be 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 can be optimized as described in the paper "Handling Churn in a DHT" [Handling_Churn_in_a_DHT].

b. P2P system diagnostics to check the overall health of the P2P overlay network, the consumption of network bandwidth, for the presence of problem links and also to check for abusive or malicious nodes. This is not a trivial problem and has been studied in detail for content and streaming P2P overlays [Diagnostic_Framework] as well as in earlier P2PSIP documents [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 and retrieval statistics of nodes from the overlay . The simple connectivity tests are not dependent on the type of P2PSIP overlay. Note 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 bandwidth of the link connecting the user to the Internet.

4. Overview of operations

The diagnostic mechanisms described in this document are mainly intended to detect and localize failures or monitor performance in P2PSIP overlay networks. 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 a specified peer, a mechanism to detect the availability of specified resource records and a mechanism to discover P2PSIP overlay topology and the underlay topology failures.

The P2PSIP diagnostics extensions define Inspect and Path_Track methods for connection quality check and retrieval of diagnostic information, and the Error response to these methods. Essentially it reuses P2PSIP base protocol specification and extends them to introduce the new diagnostics methods. The extensions strictly follow the P2PSIP base protocol specification on the messages routing, transporting and NAT traversal etc. The diagnostic methods are however P2PSIP protocol independent.

This document mainly describes how to detect and localize failures including disabled peers, congested peers, misrouting behaviors and underlying network faults in P2PSIP overlay networks through a simple and efficient mechanism. This mechanism is modeled after the ping/ traceroute paradigm: ping (RFC792 ICMP echo request [RFC0792]) 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 (by defining the Path_Track method) for diagnosing P2PSIP overlay networks.

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 Path_Track to determine where the fault lies.

The authors earlier considered an approach where a response was generated by each intermediate peer as the message traversed the overlay, but this approach was discarded as a result of working group discussion. One reason this approach was discarded was that it could provide a DoS mechanism, whereby an attacker could send an arbitrary message claiming to be from a spoofed "sender" the real sender wished to attack. As a result of sending this one message, many messages would be generated and sent back to the spoofed "sender" -- one from each intermediate peer on the message path. While authentication mechanisms could reduce some of the risk of this attack, it still resulted in a fundamental break from the request-response nature of the RELOAD protocol, as multiple responses are generated to a single request. Although one request with responses from all the peers in the route will be more efficient.

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.

4.1. Inspect: "Ping" behavior

To provide "ping" like behavior, 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.

4.2. Path_Track: "Traceroute" behavior

A simple Path_Track method is used 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 through symmetric routing) to get the next hop node C information and diagnostic information of B. Unless a failure prevents the message from being forwarded, this step can be iterative until the request reaches the responsible node D for the destination ID, and to retrieve diagnostic information for node D.

One application of these tools is to detect and diagnose the connectivity to the specified peer or the availability of the specified resource-record through P2PSIP Inspect operation after the overlay network receives some alarms about overlay service degradation or interruption. If the Inspect fails, one can then send a P2PSIP Path_Track to determine where the fault lies.

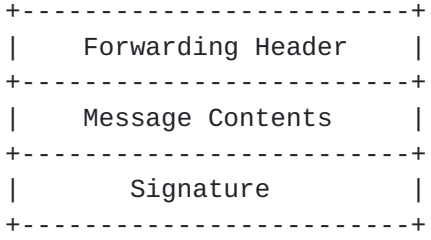
4.3. Authorization

The diagnostic information must be only be 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 for each piece of 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. RELOAD diagnostic extensions

This document extends the P2PSIP base protocol to carry diagnostics information. Considering the special usage of diagnostics, this document defines simple new methods: Inspect and Path_Track . Additionally, the related Error codes for these methods, and some useful diagnostics information are defined. Processing of the messages is discussed.

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



5.1. Message Code Extension

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 introduces two types of messages and their responses:

Name		Message Code
Inspect	request	101
Inspect	response	102
Path_Track	request	103
Path_Track	response	104

The final message code will be assigned by IANA as specified in section 13.6 of [I-D.ietf-p2psip-base].

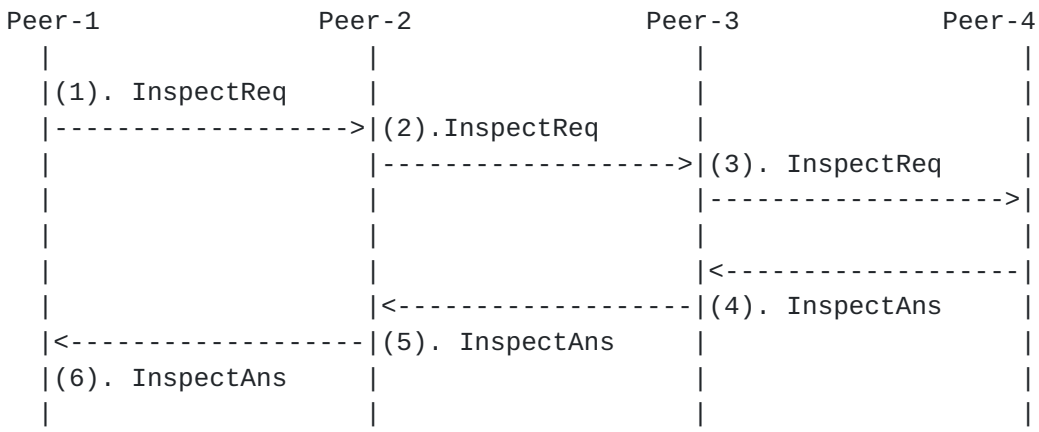
5.2. Message Type Extensions

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 to detect and localize failures in P2PSIP overlay network. The Error Codes to these requests are defined in Section 5.3.1 of this spec.

5.2.1. 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 hop counter. Here we define a new method Inspect for connectivity quality check purposes.



Inspect example

See below for the Inspect formats.

```
Inspect Request:
         struct {
           uint64 expiration;
           uint64 timestampInitiated;
         } InspectReq;

Inspect Response:
         struct {
           uint64 expiration;
           uint8  hopCounter;
           uint64 timestampReceived;
         }InspectAns;
```

expiration : The time-of-day (in seconds and microseconds, according to the receiver's clock) in NTP timestamp format [RFC4330] when the Inspect request expires. This field can be used to mitigate the replay attack to the destination peer and overlay network.

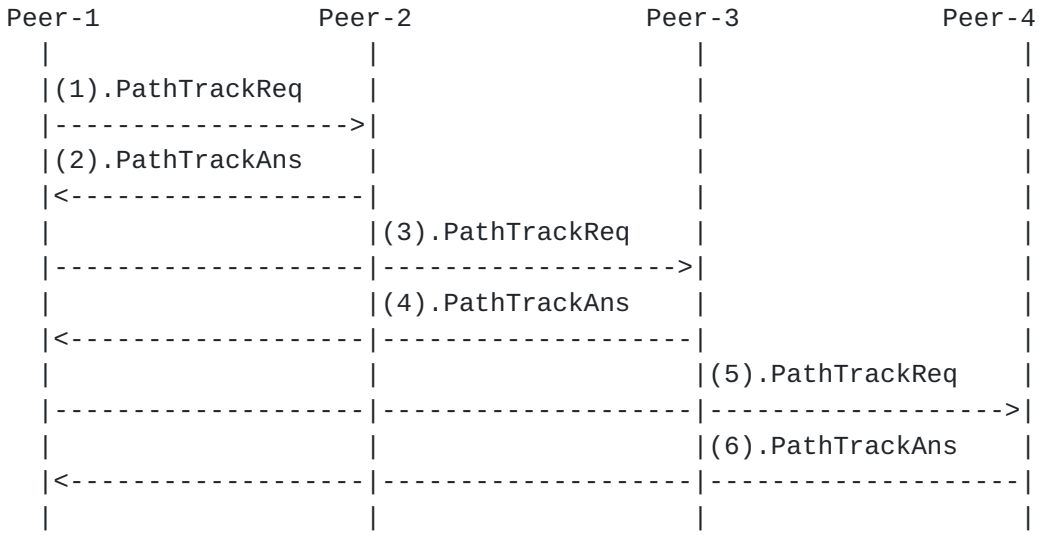
timestampInitiated : The time-of-day (in seconds and microseconds, according to the sender's clock) in NTP timestamp format [RFC4330] when the P2PSIP Overlay diagnostic request is sent.

timestampReceived : The time-of-day (in seconds and microseconds, according to the receiver's clock) in NTP timestamp format [RFC4330] when the P2PSIP Overlay diagnostic request was received.

hopCounter : This field only appears in diagnostic responses. It must be exactly copied from the TTL field of the forwarding header in the received request. This information is sent back to the request initiator, allowing it to compute the hops that the message traversed in the overlay.

5.2.2. Path_Track

This document defines 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 its status information like congestion state, its processing power, its available bandwidth, the number of entries in its neighbor table, its uptime, its identity and network address information, and the next hop peer information.



Path_Track example

A Path_Track request specifies which diagnostic information is requested by setting different bits in the flag contained in the Path_Track request. If the flag is clear (no bits are set), then the Path_Track request is only used for requesting the next hop information. In this case the iterative mode of Path_Track is degraded to a Route_Query method which is only used 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 routing mode chosen by the initiator node.

A response to a successful PathTrackReq is a PathTrackAns message. There is a general diagnostic information portion of the payload, the contents of which are based on the flags in the request. Please refer to Section 5.3.2 for the definitions of the diagnostic information.

```
Path_Track request:
          struct {
            Destination           destination;
            uint64                expiration;
            uint64                timestampInitiated;
            uint8                 length;

            select (length){
            case 0:
              uint64                dMFlags;

            case > 0:
              uint64                dMFlags;
              uint64                dEFlags<0...length-1>;
            }
          } PathTrackReq;
```

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.

expiration : The time-of-day (in seconds and microseconds, according to the receiver's clock) in NTP timestamp format [RFC4330] when the Path_Track request expires. This field can be used to mitigate the replay attack to the destination peer and overlay network.

timestampInitiated : 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.

length : the number of extended diagnostics flags (in the unit of 64 bits). If the value is greater than or equal to 1, then one or more extended diagnostics flags (dEFlags) are specified. The value of length must not be negative.

dMFlags : A mandatory flag which is an unsigned 64-bit integer indicating which kind of diagnostic information the initiator is interested in. The initiator sets different bits to retrieve different kinds of diagnostic information. If dMFlags is clear, then no mandatory diagnostic information is conveyed in the Path_Track response. If dMFlag is set to all '1's, then all diagnostic information kinds are requested. (Note: This memo specifies the initial set of flags, the flags can be extended by standard action We will add a section about extending the flags both standard and application specific in a future version) The dMflags indicate general diagnostic information The mapping between the bits in the dMFlags and the diagnostic information

kind presented is as below.

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

ROUTING_TABLE_SIZE(0x0002) : if set, the number of entries in the responding peer's neighbor is requested;

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

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

SOFTWARE_VERSION(0x0010): if set, the software version of the peer program is requested;

MACHINE_UPTIME(0x0020): if set, the uptime of the machine is requested;

APP_UPTIME(0x0040): if set, the uptime of the p2p application is requested;

MEMORY_FOOTPRINT(0x0080): if set, the memory footprint of the peer program is requested;

DATASIZE_STORED(0x0100): if set, the number of bytes of data being stored by this node is requested;

MESSAGES_SENT_RCVD(0x0200): if set, an array element containing the number of messages sent and received is requested;

EWMA_BYTES_SENT(0x0400): if set, an integer representing the exponential weighted average of bytes sent per second by this peer is requested;

EWMA_BYTES_RCVD(0x0800): if set, an integer representing the exponential weighted average of bytes received per second by this peer is requested;

UNDERLAY_HOP (0x1000): if set, a byte representing the IP layer hops to the next hop (P2P overlay layer) of this message is requested;

BATTERY_STATUS(0x2000): if set, a byte representing whether the peer device is using battery power or not is requested;

dEFlags : the extended diagnostics flags which can be used by

applications to retrieve its own customized diagnostics information. This allows private extensions.

Path_Track response:

```
          struct {
                Destination     next_hop;
                uint64          expiration;
                uint64          timestampReceived;
                uint8           length;
                Diagnostic_Info diag_info_list<0..length-1>;
          } PathTrackAns;
```

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.

expiration : The time-of-day (in seconds and microseconds, according to the receiver's clock) in NTP timestamp format [RFC4330] when the Path_Track response expires. This field can be used to mitigate the replay attack to the destination peer and overlay network.

timestampReceived : The time-of-day (in seconds and microseconds, according to the receiver's clock) in NTP timestamp format [RFC4330] when the P2PSIP Overlay diagnostic request was received.

length : the number of Diagnostic_Info values contained in the response.

diag_info_list : The diagnostic information from the responding peer.

The TLV structure for Diagnostic_Info is as follows:

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

kind : A numeric code indicating the type of information being reported.

length : the length in bytes of the opaque date containing the information being reported

diagnostic_information : Data of length specified above containing the value for the diagnostic information being reported.

Various kinds of diagnostic information can be retrieved, Please refer to section Section 5.3.2 for details of the types and Kind-ID for the diagnostic information that may be reported.

5.3. Message Payload Extensions

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.

5.3.1. Error Codes

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

Name	Message Code
Error	0xFFFF

This document defines new Error codes to carry different failure reports to the initiator node when failure is detected during diagnostics. This document introduces new Error Codes as below:

Code Value	Error Code Name
101	Underlay Destination Unreachable
102	Underlay Time exceeded
103	Message Expired
104	Upstream Misrouting
105	Loop detected
106	TTL hops exceeded

The final error codes will be assigned by IANA as specified in section 13.7 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 101 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 more human readable.

5.3.2. Diagnostics information

This document introduces some diagnostics information conveyed in the message payload which can be retrieved to get the statistics and also allows for retrieval of other kinds that a node stores. In essence, the usage allows querying a node's state such as storage and network to obtain the relevant information. It can also be used to discover information such as the software version, uptime, routing table, stored resource-objects, performance statistics of a peer and link quality of a overlay route. The diagnostic information data kinds are defined below.

PROCESS_POWER (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.

Editor's note: For the diagnostic information of processing power, bandwidth and etc., we should look at what has been useful for PlanetLab and in commercial deployments in this context, and further discussion is needed on what mature diagnostics information for p2p overlays can be brought here.

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.

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

An usage of STATUS_INFO is for congestion-aware routing. In this scenario, each peer has to update its congestion status periodically, an intermediate peer in the DHT network will choose its next hop according to both the DHT routing algorithm and the status information, and then forward requests to the chosen next hop, so as to avoid increasing load on congested peers.

SOFTWARE_VERSION: A single value element containing a US-ASCII string that identifies the manufacture, model, and version of the software.

MACHINE_UPTIME (64 bits): A single value element containing an unsigned 64-bit integer specifying the time the nodes has been up in seconds.

APP_UPTIME (64 bits): A single value element containing an unsigned 64-bit integer specifying the time the p2p application has been up in seconds.

MEMORY_FOOTPRINT (32 bits): A single value element containing an unsigned 32- bit integer representing the memory footprint of the peer program in kilo bytes.

Note: A kilo byte in this document represents 1024 bytes.

DATASIZE_STORED (64 bits): An unsigned 64-bit integer representing the number of bytes of data being stored by this node.

INSTANCES_STORED: An array element containing the number of instances of each kind stored. The array is index by Kind-ID. Each entry is an unsigned 64-bit integer.

MESSAGES_SENT_RCVD: An array element containing the number of messages sent and received. The array is indexed by method code. Each entry in the array is a pair of unsigned 64-bit integers (packed end to end) representing sent and received.

EWMA_BYTES_SENT (32 bits): A single value element containing an unsigned 32-bit integer representing an exponential weighted average of bytes sent per second by this peer. sent = alpha x sent_present + (1 - alpha) x sent where sent_present represents the bytes sent per second since the last calculation and sent represents the last calculation of bytes sent per second. A suitable value for alpha is 0.8. This value is calculated every five seconds.

EWMA_BYTES_RCVD (32 bits): A single value element containing an unsigned 32-bit integer representing an exponential weighted average of bytes received per second by this peer. Same calculation as above.

UNDERLAY_HOP (8 bits): It indicates the IP layer hops from the intermediate peer which receives the diagnostics message to its next hop peer for this message.

Note: this is from the underlayTTL in the previous version. However, RELOAD does not require the intermediate peers to look into the message body. So here we use Path_Track to gather underlay hops for diagnostics purpose.

BATTERY_STATUS (8 bits): The left-most bit is used to indicate whether this peer is using battery or not. If this bit is clear ('0'), then the peer is using battery power. The other 7 bits are to be determined by specific applications.

Editor's Note: The self-tunning draft [I-D.maenpaa-p2psip-self-tuning] could extend the diagnostics information here to collect related information for calculating self-tunning parameters.

5.4. Message Processing

5.4.1. Message Creation and Transmission

When constructing either an Inspect message or a Path_Track message, the sender MUST set both the destination and the expiration value (in NTP timestamp format) for the message.

When constructing an Inspect message, the sender MAY specfify a value for underlayTTL. If a value is not specified, the sender MUST set underlayTTL to 0. The sender MUST generate an NTP format timestamp for the current time of day and place it in the timeStampInitiated field.

When constructing a Path_Track message, the sender MUST set the

length value to a value equal to or greater than 0. If the value of length is set to 0, the sender MUST include a dMFlags value, and MUST NOT include any dEFlags values. If the sender wishes to send dEFlags in addition to dMFlags, the sender MUST set the value of length to be equal to the number of dEFlags present. Note that the sender MUST NOT set length to negative. The sender MAY set bits in dMFlags as discussed above to request specific information, and MAY set bits in dEFlags fields to request user-specific bits. The sender also MAY set dMFlags to all zero, indicating that no diagnostic information is requested.

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 degraded to a Route_Query method which is only used 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 routing mode chosen by the initiator node.

5.4.2. Message Processing: Intermediate Peers

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 MUST continue process this request according to the overlay specified routing mode from the base draft.

In p2psip overlay, the error response can be generated by the intermediate peer or responsible peer, to a diagnostic message or other messages. When a request is received at a peer, the peer may find some connectivity failures or malfunction peers through the pre-defined rules of the overlay network, e.g. by analyzing 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. All error responses contain the Error code followed by the subcode and descriptions if existed.

Each intermediate peer receiving an Inspect or Path_Track 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 with Error Code 103 "Message Expired" and return it to the initiator node, and discard the message.

The peer should return an Error response with the Error Code 101 "Underlay Destination Unreachable" when it receives an ICMP message with "Destination Unreachable" information after forwarding the

received request to the destination peer.

The peer should return an Error response with the Error Code 102 "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 Error Code 104 "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 105 "Loop detected" when it finds a loop through the analysis of via list.

The peer should return an Error response with Error Code 106 "TTL hops exceeded" when it finds that the TTL field value is no more than 0 when forwarding.

With Path_Track, if a former Path_Track message does not arrive at the destination, then the following Path_Track request must copy the next_hop field in the former response into the forwarding header and keep the destination_ID unchanged.

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.

5.4.3. Message Response Creation

When a diagnostic request message arrives at a peer, and it is responsible for the destination ID specified in the forwarding header, it MUST follow the specifications defined in 5.1.3 of the base draft to form the response header.

When the responsible peer receives an Inspect or Path_Track request/ response it MUST check the expiration value (NTP format) to determine if the message expired. If the message expired, the peer MUST generate a message with the Error Code 103 "Message Expired" and return it to the initiator node, and discard the message.

If the request is an Inspect, the destination peer MUST copy the TTL value to the HopCounter field. The receiver MUST generate an NTP format timestamp for the current time of day and place it in the timestampReceived field.

The initiator node, as well as the responding peer, MAY compute the

overlay One-Way-Delay time through the value in timestampReceived and the timestampInitiated field. However, for a single hop measurement, the traditional measurement methods MUST be used instead of the overlay layer diagnostics methods.

Editor note: We need more discussion and careful consideration on how to use the timestamp here because time synchronization is a barrier in open Internet environment, while in the operator's network, it may be less of a problem.

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.

If the request is a Path_Track, the destination peer MUST check if the initiator node has the authority to get certain kinds of diagnostic information, and if appropriate, appends the diagnostic information requested in the dEFlags and dMFlags to the response message. The peer should return an Error response with the Error Code 1 "Error_Unauthorized" when the initiator node does not have the authority to get the corresponding diagnostic information.

In the event of an error, an error response containing the error code followed by the subcode and description (if they exist) MUST be created and sent to the sender.

6. Examples

Below, we sketch how these metrics can be used.

6.1. Example 1

A peer may set EWMA_BYTES_SENT and WEMA_BYTES_RCVD flags in the PathTrackReq to its direct neighbors. A peer can use EWMA_BYTES_SENT and EWMA_BYTES_RCVD of another peer to infer whether it is acting as a media relay. It may then choose not to forward any requests for media relay to this peer. Similarly, among the various candidates for filling up routing table, a peer may prefer a peer with a large UPTIME value, small RTT, and small LAST_CONTACT value.

6.2. Example 2

A peer may set the StatusInfo Flag in the PathTrackReq to a remote destination peer. The overlay has its own threshold definition for congestion. The peer can get knowledge of all the status information of the intermediate peers along the path. Then it can choose other

paths to that node for the later requests.

6.3. Example 3

A peer may use Inspect to evaluate the average overlay hops to other peers by sending InspectReq to a set of random resource or node IDs in the overlay. A peer may adjust its timeout value according to the change of average overlay hops.

7. Security Considerations

The authorization for diagnostics information must be designed with care to prevent it becoming a resort to retrieve information for bots attacks. It should also be careful that attackers can use diagnostics to analyze overlay information to attack certain key peers if there are. As this draft is a RELOAD extension, it follows RELOAD message header and routing specifications, the common security considerations described in the base draft [I-D.ietf-p2psip-base] are also applicable to this draft.

8. IANA Considerations

8.1. Message Code

This document introduces two new types of message to the "RELOAD Message Code" Registry as below:

Message Code Name	Code Value	RFC
inspect_req	101	RFC-AAAA
inspect_ans	102	RFC-AAAA
path_track_req	103	RFC-AAAA
path_track_ans	104	RFC-AAAA

8.2. Error Code

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

Code Value	Error Code Name
101	Underlay Destination Unreachable
102	Underlay Time exceeded
103	Message Expired
104	Upstream Misrouting
105	Loop detected
106	TTL hops exceeded

8.3. Data Kind-ID

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

Kind	Kind-ID
StatusInfo	101
ProcessPower	102
Bandwidth	103

8.4. Diagnostics Flag

IANA SHALL create a "RELOAD Dianogsitcs Flag" Registry. Entries in this registry are 1-bit flag contained in a 64-bits long integer dMFlags denoting diagnostic information to be retrieved as described in Section 5.2.2. New entries SHALL be defined via [RFC5226] Standards Action. The initial contents of this registry are:

diagnostic information	diagnostic flag in dMFlags	RFC
Reserved	0x 0000 0000 0000 0000	RFC-BBBB
STATUS_INFO	0x 0000 0000 0000 0001	RFC-BBBB
ROUTING_TABLE_SIZE	0x 0000 0000 0000 0002	RFC-BBBB
PROCESS_POWER	0x 0000 0000 0000 0004	RFC-BBBB
BANDWIDTH	0x 0000 0000 0000 0008	RFC-BBBB
SOFTWARE_VERSION	0x 0000 0000 0000 0010	RFC-BBBB
MACHINE_UPTIME	0x 0000 0000 0000 0020	RFC-BBBB
APP_UPTIME	0x 0000 0000 0000 0040	RFC-BBBB
MEMORY_FOOTPRINT	0x 0000 0000 0000 0080	RFC-BBBB
DATASIZE_STORED	0x 0000 0000 0000 0100	RFC-BBBB
MESSAGES_SENT_RCVD	0x 0000 0000 0000 0200	RFC-BBBB
EWMA_BYTES_SENT	0x 0000 0000 0000 0400	RFC-BBBB
EWMA_BYTES_RCVD	0x 0000 0000 0000 0800	RFC-BBBB
UNDERLAY_HOP	0x 0000 0000 0000 1000	RFC-BBBB
BATTERY_STATUS	0x 0000 0000 0000 2000	RFC-BBBB
Reserved	0x FFFF FFFF FFFF FFFF	RFC-BBBB

9. 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, Jiang Haifeng, Erkki Harjula and Jouni Maenpaa for the email discussion and their valued comments, and special thanks to Henry Sinnreich for contributing to the usage scenarios text. We would like to thank the authors of the p2psip base draft for transferring text about diagnostics to this document.

The authors would also like to thank the many people of the IETF P2PSIP WG that have contributed to discussions and provided input invaluable in assembling this document.

10. Appendix: Changes

10.1. Changes since draft-ietf-p2psip-diagnostics-00

1. Change Title from "Diagnose P2PSIP Overlay Network" into "P2PSIP Overlay Diagnostics";

2. Change the table of contents. Add a section about message processing and another section about examples;

3. Merge diagnostics text from the p2psip base draft 01;

4. Remove ECHO method for security considerations.

10.2. Changes since 01 version

1. Add BATTERY_STATUS as diagnostic information

2. Remove UnderlayTTL test from the Inspect method, instead adding an UNDERLAY_HOP diagnostic information for PathTrack method.

3. Give some examples for diagnostic information, and give some editor's notes for futher work.

11. References

11.1. Normative References

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[I-D.ietf-p2psip-sip] Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., and H. Schulzrinne, "A SIP Usage for RELOAD", draft-ietf-p2psip-sip-03 (work in progress), October 2009.

[I-D.ietf-p2psip-base] Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol", draft-ietf-p2psip-base-06 (work in progress), November 2009.

[I-D.zheng-p2psip-diagnose] Yongchao, S. and X. Jiang, "Diagnose P2PSIP Overlay Network", draft-zheng-p2psip-diagnose-04 (work in progress), December 2008.

[Overlay-Failure-Detection] Zhuang, S., "On failure detection algorithms in overlay networks", Proc. IEEE Infocomm, Mar 2005.

[Handling_Churn_in_a_DHT] Rhea, S., "Handling Churn in a DHT", USENIX Annual Conference, June 2004.

[Diagnostic_Framework] Jin, X., "A Diagnostic Framework for Peer-to-Peer Streaming", 2005.

[Diagnostics_and_NAT_traversal_in_P2PP] Gupta, G., "Diagnostics and NAT Traversal in P2PP - Design and Implementation", Columbia University Report , June 2008.

11.2. Informative References

[RFC4330] Mills, D., "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI", RFC 4330, January 2006.

[RFC4981] Risson, J. and T. Moors, "Survey of Research towards Robust Peer-to-Peer Networks: Search Methods", RFC 4981, September 2007.

[I-D.ietf-behave-rfc3489bis] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for (NAT) (STUN)", draft-ietf-behave-rfc3489bis-18 (work in progress), July 2008.

[I-D.matuszewski-p2psip-security-requirements] Yongchao, S., Matuszewski, M., and D. York, "P2PSIP Security Overview and Risk Analysis", draft-matuszewski-p2psip-security-requirements-06 (work in progress), September 2009.

[I-D.song-p2psip-security-eval] Yongchao, S., Zhao, B., Jiang, X., and J. Haifeng, "P2PSIP Security Analysis and Evaluation", draft-song-p2psip-security-eval-00 (work in progress), February 2008.

[I-D.baset-p2psip-p2pp] Baset, S., Schulzrinne, H., and M. Matuszewski, "Peer-to-Peer Protocol (P2PP)", draft-baset-p2psip-p2pp-01 (work in progress), November 2007.

[I-D.ietf-mmusic-ice] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-ietf-mmusic-ice-19 (work in progress), October 2007.

[I-D.bryan-p2psip-app-scenarios] Bryan, D., Shim, E., Lowekamp, B., and S. Dawkins, "Application Scenarios for Peer-to-Peer Session Initiation Protocol (P2PSIP)", draft-bryan-p2psip-app-scenarios-00 (work in progress), November 2007.

[I-D.bryan-p2psip-requirements] Bryan, D., "P2PSIP Protocol Framework and Requirements", draft-bryan-p2psip-requirements-00 (work in progress), July 2007.

[I-D.maenpaa-p2psip-self-tuning]
Maenpaa, J., Camarillo, G., and J. Hautakorpi, "A Self-tuning Distributed Hash Table (DHT) for REsource LOcation And Discovery (RELOAD)", draft-maenpaa-p2psip-self-tuning-01 (work in progress), October 2009.

[I-D.ietf-p2psip-concepts]
Bryan, D., Matthews, P., Shim, E., Willis, D., and S. Dawkins, "Concepts and Terminology for Peer to Peer SIP", draft-ietf-p2psip-concepts-02 (work in progress), July 2008.

Authors' Addresses

Song Haibin
Huawei
Baixia Road No. 91
Nanjing, Jiangsu Province 210001
P.R.China

Phone: +86-25-84565867
Fax: +86-25-84565888
Email: melodysong@huawei.com

Jiang Xingfeng
Huawei
Baixia Road No. 91
Nanjing, Jiangsu Province 210001
P.R.China

Phone: +86-25-84565868
Fax: +86-25-84565888
Email: jiang.x.f@huawei.com

Roni Even
Huawei
14 David Hamelech
Tel Aviv 64953
Israel

Email: even.roni@huawei.com

David A. Bryan
Cogent Force, LLC
Williamsburg, Virginia
United States of America

Email: dbryan@ethernot.org