P2PSIP Working Group J. Peng
Internet-Draft L. Deng
Intended status: Standards Track L. Le
Expires: August 20, 2013 G. Li
China Mobile
X. Ma
Beijing University of Posts and
Telecommunications
Feburary 16, 2013

Proposals for RELOAD to support Promotion and Demotion for User-owned Nodes
draft-peng-p2psip-promotion-02

Abstract

This document proposes extensions to RELOAD to support flexible client promotion and demotion modes. RELOAD aims at providing a uniform protocol for both overlay clients and peers, where promotion of a client to peer is triggered and completed at the client's pleasure. It is proposed that RELOAD provide a more restrictive framework to enable passive promotion and demotion, where decisions are made by the network rather than individual user-owned nodes.

Status of this Memo

This Internet-Draft will expire on August 20, 2013.

Copyright Notice

Table of Contents

1. Introduction

RELOAD[I-D.ietf-p2psip-base], a peer-to-peer (P2P) signaling protocol for use on the Internet, provides a generic, self-organizing overlay network service, allowing nodes to efficiently route messages to other nodes and to efficiently store and retrieve data in the overlay.

There are two types of roles in the RELOAD architecture: peer and client. Clients are merely users of the basic messaging and storage function provided by the overlay, while peers (i.e. non-client nodes) are active participants of the serving overlay as they collaborate in a P2P manner to serve one another.

For Internet services on the basis of a RELOAD-enabled P2P overlay, it is appealing to exploit the collectively abundant resources of UNs (i.e. user-owned nodes) to further reduce service operator's CAPEX (i.e. capital expenses on dedicated serving equipment), thus indicating an application scenario for on-demand passive client promotion. On the other hand, due to the distributed nature of P2P overlays, undesirable implications often arises after imprudent peer exits, demanding for regulated passive peer demotion to client in reverse.

However, unlike normal clients in RELOAD overlay, individual UNs are featured with more restrictive resource limits, considerable capability heterogeneity, and diverse interest groups, whose promotion/demotion demands for more restrictive regulations than the simple active client promotion facility defined in RELOAD.

A SIP usage could be used to illustrate the UN-oriented promotion/ demotion problem, where a UN-oriented client refers to a user-owned node attached originally for SIP UAs, while a peer refers to a dedicated SIP servers or UN-promoted SIP servants of the RELOAD overlay.

In this draft, the basic problems for promoting/demoting User-owned clients to/from serving peers using the currently available mechanism are outlined from the overlay operator's point of view, followed by a detailed analysis of specific functionality requirements. Potential extensions to RELOAD are summarized in Section 5. Relevant security considerations are stated in Section 6.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Problem Statement

3.1. Passive promotion instead of active promotion

Since a UN generally lacks the incentive to serve others proactively as a result of its relatively bounded resources in combination with the autonomous and self-centered nature of its individual owner, the UN-oriented promotion procedures are expected to be largely triggered from the overlay network side for the benefit of the servicing system as a whole. Restrictive regulations are needed in the passive procedure for UN-oriented client promotion. Firstly, a UN may be incapable of or malfunctioning in serving others. Secondly, a UN may be preferred not to become a peer for the sake of network considerations, e.g. a moderate network size may be preferred to remain efficient overlay routing.

The simple join-update method allows for active promotion only, where an successfully attached client (either UN-oriented or not), spontaneously initiates the promotion procedure by sending JoinReq messages to expected overlay neighboring peers and indicates its ready-to-go status as a serving peer by sending UpdateReq messages with revised routing information.

It is therefore proposed that the promoter (e.g. an overloading peer) rather than the UN in question triggers the correspondent passive promotion procedure, and the promoted UN is selected from a potentially large client group of candidates and certified by the promoter to other peers in the overlay to be recognized as an authorized peer.

Potential extensions to RELOAD include extensions to certain kinds of messages (e.g. Probe and Join) in order to support the passive client promotion procedure. To see more details in the next section.

3.2. Passive demotion as well as active demotion

For a successfully promoted UN peer, one of the following two demotion scenarios would ultimately terminates its current peering service in the RELOAD overlay.

o Active demotion decision from the UN side, that the user/UN decides to cease being a peer, for instance:

  * Case 1: if the user perceives a considerable decrease in user experience; or

  * Case 2: if the UN terminal runs out of battery.

- Passive demotion decision from the overlay side to get rid of unnecessary UN-promoted peers, for instance:

  * Case 3: Overlay decides to shrinks its peer group to maintain a tolerable routing delay;

  * Case 4: Overlay decides to exclude the peer(s) for unsatisfactory service provision.

The current leave-without-response method fits well in the active demotion scenarios, where a peer (either promoted earlier from UN-oriented client or not), spontaneously initiates the demotion procedure by sending LeaveReq messages to its overlay neighboring peers and be free to go without any further confirmation. This simple but kind of abrupt mode is ungracefull in comparison with another mode of peer demotion mode (i.e. graceful mode, where the demoting peer takes an active part in the data migration and routing update for the resultant overlay adaptation before its physical exit.

It is therefore proposed to add support to allow for both active and passive peer demotion procedures for UN-oriented peers. Moreover, graceful demotion mode is highly preferable for passive demotion scenarios.

Potential extensions to RELOAD include extensions to current method (e.g. extension to LeaveReq message or may be introduction of new types of messages such as explicit LeaveRes messages) to support peers graceful demotion mode. To see more details in the next section.

4. Extensions to RELOAD

In this section, we introduce extensions to RELOAD to enable restrictive promotion and demotion.

4.1. Configuration file

It would be better to have an announcement about promotion-related extensions to the configuration file.

A new label should be defined like below.

```
<clientpromotion-permitted> true </clientpromotion-permitted >
```

This element represents whether clients in the overlay can be promoted, and be defaulted as "true" when absent.

It is desirable to make the "right" client to be promoted based on the observation of its expected serving capability. In other words, one should get some local capability statistics about promoting candidate during the procedure of promotion. It is preferred that they refer to the same measuring tools in order to get the statistics (e.g. client CPU, Memory or Disk capabilities) for the same standard. While this is easy to do with memory and disk (in MB for instance), it is not the case with CPU. Hence another new label should be defined here which offer the URL at which the common measuring tool for clients' CPU capability can be downloaded like below:

```
<benchmark-location> http://example_for_cpu-benchmark.com:82/download.rar </
benchmark-location>
```

4.2. Probe

4.2.1. ProbeInformationType

```
 enum{reservedProbeInformation(0),
   responsible_set(1), num_resources(2), uptime(3),
   client_capability(4), peer_demotion (5)  (255)} ProbeInformationType;
```

The ProbeInformationType gives an enumeration of information type which the requester would like the responder to provide. The first four parameters have already been defined in [draft-ietf-p2psip-base], and the last one is a new parameter defined here which is important in promotion procedure.

responsible_set

It indicates that the peer should respond with the fraction of the overlay for which the responding peer is responsible.

num_resources

It indicates that the peer should respond with the number of resources currently being stored by the peer.

uptime

It indicates that the peer should respond with how long the peer has been up in seconds.

client_capability

It indicates that the client should respond with a list of values which stands for its capability. For it is the main item to be considered in promotion. The values include the capability of CPU, Memory, Disk and so on. These values form an array, different integer index stands for different client's capability. For example integer 1 equals CPU while integer 2 stands for Memory. While the available memory and disk may be termed as quantitative values easily, CPU capabilities may not. However, an overlay may specify a benchmark in the configuration file for its participants to be used for measuring its CPU's capability as quantitative values.

peer_demotion

This structure indicates that a peer may want another peer to be demoted after considering the situation of the overlay, which means a kind of passive demotion of peers. The response to this structure includes an array of Boolean values collected by the peer which is to be demoted. The array of values shows different responses to the peer's demotion, and these responses are from the peers which are direct successors to the peer to be demoted.

4.2.2. ProbeReq message

```
struct {
  probeInformationType     requested_info<0..2^8-1>;
} ProbeReq
```

The structure of ProbeReq gives a list of the piece of status information that the requester want to get.

4.2.3. ProbeAns message

```
typedef uint32<0...n>      client_capability_list;
typedef Boolean<0...n>     demotion_response_list;

struct {
  select (type) {
    case responsible_set:
       uint32           responsible_set;
    case num_resources:
       uint32           num_resources;
    case uptime:
       uint32           uptime;
    case client_capability:
       client_capability_list     capability_list;
    case peer_demotion:
       demotion_response_list     response_list;
  };
} ProbeInformationData;
```

The structure of ProbeInformationData is an item of the ProbeInformation message. It gives the value of related type.

```
struct {
  ProbeInformationType    type;
  uint8                   length;
  ProbeInformationData    value;
} ProbeInformation;
```

The structure of ProbeInformation gives the type and value of a probe item. What's more, it also contains the length of this message.

```
struct {
  ProbeInformation        probe_info<0..2^16-1>;
} ProbeAns;
```

This message is the response to ProbeReq, and the variable ProbeInformation is just related to variable ProbeInformationType in the structure ProbeReq.

For example, if a ProbeReq message contains a type of client_capability, its response ProbeAns must have a value of the client_capability. OverloadedPeer will use this pair of messages to collect information about the clients connecting to it, so that it can make a decision on which client to promote.

Moreover, if a ProbeReq message contains a type of peer_demotion, its response ProbeAns must have a value of the demotion_response_list. In this value there are responses from the neighbor of the peer which may be demoted. These responses have an influence on whether the demotion will continue.

5. Update

5.1. Update_type

```
enum { reservedevent(0),ue_promotion(1), peer_demotion(2), routing_table(3),
(255)}
Update_Type;
```

Update_Type defines three kinds of update event here. Parameter Ue_promotion means the promotion of client and peer_demotion indicates the demotion of peers while parameter routing_table stands for updating routingtables. The parameter reservedevent(0) means other extensions that can be made to this structure if needed. And it is useful for other Topology Plugins to make extension with this parameter. Even more important, different Topology Plugins can connect to RELOAD protocol stack through this extension mechanism, and they can also decide whether it is needed to deal with the received Update message through the Update_Type in the message. In this way, different Topology Plugins can work together to some degree.

5.2. SecutrityBlock

```
   Struct{
     CertificateType      type;
     Opaque             certificate<0..2&#65342;16-1>;
   }GenericCertificate;


   Struct{
     GenericCertificat   certificates<0..2&#65342;16-1>;
     Signature          signature;
   }SecurityBlock;
```

The two above structures have already been defined in [draft-ietf-p2psip-base]. Maybe here it will be desired to define a new CertificateType, because it is generated by Overloaded Peer which is different from that generated by ES (enrollment server). What's more, they are the items of message UpdateInformation.

5.3. UpdateInformation structure

```
   struct{
     select (update_type) {
       case  ue_promotion:
         NodeId            Overloaded_Peer_id;
         NodeId            Promotion_Client_id;
         NodeId            expect-peer-id-top;
         NodeId            expect-peer-id-floor;
         SecurityBlock     securityblock;

       case  peer_demotion:
         NodeId            Administrating_Peer_id;
         NodeId            Demoting_Peer_id;
         SecurityBlock     securityblock;

       case  routing_table:
         RoutingTable_List     routing_table_list <0...n>;
     }
   } UpdateInformation
```

Here is the value of arranged UpdateType in the message UpdateReq.

For the UpdateType ue_promotion, Overloaded_Peer_id offers the peer_id which is overloaded. And Promotion_Client_id gives the Client_id which is going to be promoted. When the client is in the procedure of promotion, it may have to change to another peer_id. If the client needs to do so, Expected_Peer_id can help to achieve it.

For the UpdateType peer_demotion, Administrating_Peer_id indicates the peer_id which initiates the command of demotion in the procedure of passive demotion. And Demoting_Peer_id tells the peer_id which is to be demoted.

The securityblock in the above two types can prevent malicious promotion or demotion to some degree.

For the UpdateType routing_table, it contains information about routing_table which has already been defined in [draft-ietf-p2psip-base], and exchanging routing_table is the main function of Update message. And more details can be referred to that draft.

5.4. UpdateReq structure

```
struct {
  UpdateType            update_type;
  uint32                length;
  UpdateInformation     Value;
} UpdateReq;
```

The structure of UpdateReq gives what kind of update it will do, and the information which is needed to achieve that. And the length of this message is also added.

5.5. UpdateAns structure

```
struct {
  uint32       update_response;
}UpdateAns;
```

The structure of UpdateAns is response to the UpdateReq message. The variable update_response may stand for success, fail, error and so on.

6. Leave

6.1. LeaveType

```
enum {reserved(0), Node_Leave(1), Peer_Demotion(2), (255)}
LeaveType
```

The LeaveType gives two kinds of coiditions. Firstly, one node is leaving the overlay; secondly, one peer is in the procedure of demotion no matter it is active or passive.

6.2. LeaveReq

```
struct {
  leaveType             type;
  select (type){
    case  Node_Leave:
      NodeId            leaving_node_id;
    case  Peer_Demotion:
      NodeId            demoting_peer_id;
  };
} LeaveReq;
```

The sturucture of LeaveReq contains the kind of leaving event and related node_id. When the type is Node_Leave, the variable leaveing_node_id may be a client_id or a peer_id, which means the node is leaving. But when the type is Peer_Demotion, the variable demoting_peer_id must be a peer_id. But in the procedure of demotion, there are two options. The client continues to use its peer_id before demotion or changes to the original client_id which is generated by ES (enrollment server). The former option is preferred, for the consideration that an earlier overloaded peer is more likely to get overloaded again in the near future than other peers. Hence it would be desirable to keep some highly capable clients available around these "vulnerable" peers, and be at hand for potential promotion operations. Moreover, one may expect this client migration to be an effective case for the population evolution for the clients as well as peers in the network, leading to a more load-balanced manner.

6.3. LeaveAns

```
   struct{
     LeaveType             type;
     select (type){
       case  Node_Leave:
         uint32            leave_response;
       case  Peer_Demotion:
         uint32            demotion_response;
     };
   } LeaveAns;
```

The structure of LeaveAns is response to the LeaveReq message. For both of the LeaveType, the variable leave_response and demotion_response may stand for success, fail, error and so on.

7. Message Flow

In this section, three message flows are given respectively for passive promotion, active demotion and passive demotion of a UN.

7.1. Passive promotion

```
   PC=Promoting Client             OP=Overloaded Peer
         PC                          OP
          |                          |
          |                          |
          |ProbReq                   |
          |<-------------------------|
          |                          |
          |ProbAns                   |
          |------------------------->|
          |                          |
          |                          |
          |                          |
          |                          |
          |UpdateReq                 |
          |<-------------------------|
          |                          |
          |                          |
          |UpdateAns                 |
          |------------------------->|
          |                          |
          |                          |
          |                          |
```

(1) After finding that it is overloaded, a peer sends ProbReq messages of type "client_capability" to the client directly connected to it in order to collect some information about these clients' capability.

(2) Receiving the ProbReq messages, a willing client returns a ProbAns message which reports the current available local resources (e.g. CPU, Memory or Disk capabilities) for serving as a peer if promoted.

(3) On the basis of the collected information, the overloaded peer selects the most appropriate client and sends an UpdateReq message of type "peer_promotion"to it informing the promotion decision to the selected client, along with correspondent delegation information (e.g. delegation certificate, UpdateType, UpdateInformation).

(4) The selected client returns UpdateAns message to acknowledge the command reception and initializes the procedure of acquiring the expected node_id from ES and joining the network again as a peer.

7.2. Active demotion

```
   DP=Demoting Peer   PPs=Previous Peers    NPs=Next Peers
       DP               PPs                NPs
        |                |                  |
        |                |                  |
        |LeaveReq        |                  |
        |--------------->|                  |
        |                |                  |
        |LeaveReq        |                  |
        |---------------------------------->|
        |                |                  |
        |LeaveAns        |                  |
        |<---------------|                  |
        |                |                  |
        |LeaveAns        |                  |
        |<----------------------------------|
        |                |                  |
        |StoreReq        |                  |
        |---------------------------------->|
        |                |                  |
        |StoreAns        |                  |
        |<----------------------------------|
        |                |                  |
        |UpdateReq       |                  |
        |--------------->|                  |
        |                |                  |
        |UpdateReq       |                  |
        |---------------------------------->|
        |                |                  |
        |UpdateAns       |                  |
        |<---------------|                  |
        |                |                  |
        |UpdateAns       |                  |
        |<----------------------------------|
        |                                   |
```

(1) A peer, decided to be demoted for some reason, sends LeaveReq messages to the nodes directly connected to it including its successors, predecessor and clients.

(2) The peers which receives these messages return LeaveAns, which stand for their attitude to the demotion. Actually the demoting peer don't wait for the response.

(3) Then the peer will continue the operation of data migration. It sends StoreReq messages to its successors in order to store the data for which it is responsible before.

(4) The data recipients return StoreAns messages to acknowledge the data migration.

(5) When data migration is over, the peer sends UpdateReq messages to other peers in its routing table in order to update their routing tables.

(6) Informed peers delete the demoting peer from their routing table and return UpdateAns messages to acknowledge the update operation.

7.3. Passive demotion

In order to maintain a tolerable routing delay or some reason else, some peer may be demoted. And if they are not to do that actively, they will be made to do that. In other words, some peers which can be as a role of administrator will send orders to other peers for the sake of passive demotion.

```
   DP=Demoting Peer    OP=Overloaded Peer    NPs=Neighboring Peers
      DP                  OP                   NPs
       |                   |                    |
       |                   |                    |
       |ProbReq            |                    |
       |<------------------|                    |
       |                   |                    |
       |LeaveReq           |                    |
       |------------------------------------->|
       |                   |                    |
       |LeaveAns           |                    |
       |<-------------------------------------|
       |                   |                    |
       |ProbAns            |                    |
       |------------------>|                    |
       |                   |                    |
       |UpdateReq          |                    |
       |<------------------|                    |
    +--------------------------------------------+
    |                                            |
    | Date migration and rooting tables update   |
    |                                            |
    +--------------------------------------------+
       |UpdateAns          |                    |
       |------------------>|                    |
       |                   |                    |
```

(1) Firstly the commander sends ProbeReq messages to some peer in order find out whether it is acceptable to demote this peer.

(2) The to-be-demoted peer which receives ProbeReq messages sends LeaveReq messages to its successors, querying for their consents.

(3) The successors receiving LeaveReq messages returns LeaveAns messages whether they agree or disagree with the peer's demotion after considering their own situation taking account of the load to be migrated from the demoting peer to them after demotion.

(4) The to-be-demoted peer initiates a ProbeAns message which contains the responses it has just collected. And it returns it to the commander which sended ProbeReq messages.

(5) After receiving the ProbeAns messages, the commander makes a decision whether or not to enforce the demotion. If the demotion is to be carried out, it sends UpdateReq message to the peer in question informing it to be demoted.

(6) The informed peer continues to initialize a procedure of active demotion like the one described in the above subsection, and returns a UpdateAns message to the commander to report if the operation is successful or not.

8. Security Considerations

As stated above, the group of UNs manifests diversity in both physical capabilities and public morals in terms of serving as an overlay peer. Hence, it is reasonable to conduct explicit authorization to distinguish a promotion candidate's potential to serve as a peer from normal UN clients on one hand, and guarantee timely revocation to limit the impact of a misbehaving promoted UN-oriented peer on the other hand.

It is therefore proposed that:

- a qualified UN acquires a separate peer certificate to attest its capabilities and willingness to serve as a peer; and

- a UN-promoted peer's certificate is revoked if it fails to deliver expected performance while its client certificate remains intact.

Potential extensions to RELOAD include separate peer certification and proactive certificate revocation.

9. IANA Considerations

There are no IANA considerations associated to this memo.

10. Acknowledgements

11. References

11.1. Normative References

[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-15 (work in progress), May 2011.

11.2. Informative References

[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-03 (work in progress), October 2010.

Authors' Addresses

Jin Peng
China Mobile
Unit 2, 28 Xuanwumenxi Ave,
Xuanwu District
Beijing 100053
P.R.China

Email: Penjin@chinamobile.com

Lingli Deng
China Mobile
Unit 2, 28 Xuanwumenxi Ave,
Xuanwu District
Beijing 100053
P.R.China

Email: Denglingli@chinamobile.com

Lifeng Le
China Mobile
Unit 2, 28 Xuanwumenxi Ave,
Xuanwu District
Beijing 100053
P.R.China

Email: Lelifeng@chinamobile.com

Gang Li
China Mobile
Unit 2, 28 Xuanwumenxi Ave,
Xuanwu District
Beijing 100053
P.R.China

Email: Ligangyf@chinamobile.com

Xiao Ma
Beijing University of Posts and Telecommunications
10 Xi Tu Cheng Rd.
Haidian District
Beijing 100876
P.R.China

Email: maxiao_bupt@139.com