P2PSIP Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 2, 2012

J. Peng
L. Le
China Mobile
K. Feng
Beijing University of Posts and Telecommunications

October 30, 2011

One Hop Lookups Algorithm Plugin for RELOAD
draft-peng-p2psip-one-hop-plugin-01

Abstract

This document proposes an implementation of overlay plugin algorithm which is called one hop lookups to provide examples and references for the research of one hop based RELOAD. With the development of the real time communications, there are high demands for the improvement of routing efficiency. In the one hop algorithm, each peer maintains complete membership information which can guarantee one hop lookups to improve the routing efficiency. For the RELOAD, using the one hop lookups algorithm to construct Topology Plugin with the same RELOAD core can have a better support for VoIP applications, and the implementation of one hop lookups algorithm plugin is based on the methods provided by RELAOD.

Table of Contents

1. Introduction

RELOAD [I-D.ietf-p2psip-base] is a peer-to-peer (P2P) signaling protocol for use on the Internet. The Architecture of RELOAD has a routing layer which is called the Topology Plugin. It is responsible for implementing the specific overlay algorithm being used. In the RELOAD base protocol, the Topology Plugin is described by Chord, and we define another DHT algorithm in order to provide a high rouging efficiency which can be used in real time communications for a better performance.

The real time communication has a high demand on the routing efficiency, for example, the VoIP usage on the application level of RELOAD, it is no doubt that the routing speed or efficiency must be very important, and it depends on the looking up efficiency of the Overlay Topology. There are many kinds of structured peer-to-peer overlay network algorithms like Chord, Pastry, CAN and so on. Most of them have one in common is that they have a routing table which can maintain a small amount of routing state. But all these algorithms need nearly O (log N) steps to find one peer or resource, if there are a large number of peers in the overlay, it costs a lot time to locate peer or resource which may have a bad influence on the application level of RELOAD. The one hop lookups algorithm gives one way to maintain complete membership information at each node in order to increase the routing efficiency.

Being a kind of Topology Plugin, O (1) protocols achieve low latency lookups on small or low-churn networks because lookups take only a few hups, but incur high maintenance traffic on large or high-churn networks. On contrast, O (log N) protocols incur less maintenance traffic on large or high-churn networks but require more lookup hops in small networks. The experiment results show that the routing table maintenance bandwidth using is still acceptable when the number of overlay nodes equal 10^5 and 10^6 [SingleHopDHT]. So, weighing the advantages and disadvantages, in a relative small and stable network like a low-churn telecom core net with VoIP applications, the O (1) algorithms are very important as a Topology Plugin of RELOAD.

In the one hop algorithm, each peer maintains a complete description of system memberships, and it presents techniques for the peers to maintain this information accurately with lower costs of communications. Different one hop algorithm implementations have different ways to keep the accuracy of routing states. For example, the one hop lookups for peer-to-peer overlay [One-Hop-Lookups] uses event notification mechanism without broadcast to update all the peers' routing table during several seconds; the SandStone [SandStone] gives a way to collect update information by SPM (Super Peer Maintenance) which is called SPM assisted one hop enhancement.

The SandStone overlay is typically deployed as a two-layered DHT, including a global DHT and several regional DHT. There is at least one SPM node in each region, and once one peer's state change is detected by its neighboring peer, the neighbor will notify its local SPM, and then the SPM will disseminate the event to other SPMs, finally each SPM will broadcast this notification to all peers under the charge of it. The D1HT [D1HT] algorithm has an EDRA (Event Detection and Reporting Algorithm) to implemnt the event notification schema. In the D1HT, all the peers are ordinary, and all of them use the EDRA to make the goal of routing table maintenance. Like the D1HT, 1h-Calot [1h-Calot] is also a kind of one hop algorithms in a purely peer-to-peer environment, but it informs the event notifications by constructing a multicasting tree. Besides, there is also an algorithm called two hop lookups [TwoHopLookups] which is an improvement of the one hop lookups to support a larger size overlay. We use the one hop lookups algorithm combined with two hop lookups, D1HT and 1h-Calot to construct the RELOAD Topology Plugin because there is no SPM like peers and two layered topology in RELOAD base.

In this draft, we first give a simple overview of one hop lookups algorithm with some definitions, then fill the overlay specific data structures into the RELOAD frame and implements this algorithm with topology messages defined in the RELOAD Topology Plugin; at last, the replication and fault tolerance strategies are stated. All of the definitions and implementations based on the requirements and methods provided by RELOAD.

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. Hash functions

In this one hop lookups algorithm topology plugin, the size of the node ID and resource ID is 128 bits. The hash of an ID can be computed by SHA-1 [RFC3174] and other hash functions.

4. Peer data structure

A peer, or a node, is responsible for a particular Resource-ID which is less than or equal to its Node-ID and is greater than the Node-ID of the previous peer in the neighborhood. In order to have a high routing accuracy and efficiency, each peer must keep and maintain a

series of data structure.

In the one hop lookups algorithm, we divide the 128-bit circular identifier space into k equal contiguous intervals which are called slices. Each slice has a slice leader, which is chosen as the node that is the successor of the mid-point of the slice space (other choosing methods are list in Section X). Similarly, each slice also can be divided into u equal-sized intervals which are called units; each unit has a unit leader, which is chosen as the successor of the mid-point of the unit space. And of course every peer in the overlay should know its unit header and slice header. The slice leader and unit leader are both ordinary peers who are chosen dynamically, and they also can be configured at first. The leader choosing strategies will be stated in section 10.

In the SandStone [SandStone], there is at least one SPM node in each region, so when one peer detects the state change of overlay in its neighborhood, it will notify its local SPM as soon as possible. The local SPM then disseminate this notification to other SPMs, at last, each SPM will broadcast this update to all peers under the charge of it. The SPM is a special peer which can collect update information and do some other management tasks. The SPM does not participate in the routing procedures and it is pre-configured. On the other hand, the SPM uses the broadcast to spread the notifications. Only the bootstrap peer in RELOAD has the broadcast function. Using the broadcast is a simple and fast way to inform information, but it costs a lot of bandwidths and also may cause the notifications redundancy. Using the event based notification mechanism defined in one hop lookups can avoid redundancy in the communications: one peer will get information only from its neighbor that is one step closer to its unit leader. This implies that within a unit, information is always flowing from the unit leader to the ends of the unit. But its disadvantage is obvious: the notifications are spread too slowly which may cause the delay of overlay stabilization.

In the D1HT, all the peers are same; they use the EDRA mechanism to make sure the event can be disseminated to all peers. To disseminate the information about the events, each peer p sends up to m propagation messages at each time interval. The D1HT algorithm uses the EDRA to spread event notifications. The details of EDRA can be found in [D1HT].

In the 1h-Calot, a purely peer-to-peer architecture is preferable, in which nodes assume equal roles and share load evently. It uses overlay multicast to efficiently replicate complete O (n) routing tables on every node. And the peer in 1h-Calot also informs other peers of its arrival or departure by multicasting a notification through a tree. The details of multicasting tree construction can be

found in [1h-Calot].

In the two hop algorithm, the architecture is same as one hop algorithm with slice leaders and unit leaders. In addition, every slice leader chooses a group of its own nodes for each other slice; the group may be chosen randomly. Then the slice leader sends routing information about one group to exactly one other slice leader. The infornation about the group is then disseminated to all members of that slice as in the one hop event notification schema. After this, every node not only keeps full routing information about nodes in its own slice, but also keeps a table of k-1 (k is the number of slice) nodes that are close to it, one from every other slice.

From above, we can conclude the information peer should maintain as follows.

Routing table:

The routing table of each node keeps all of nodes' address, and then the source node can find the destination by just one hop at most if the items of routing table are right. So it is very important to keep the correctness of each node's full routing table. In order to achieve the goal, the notification of membership change events must reach every node in the overlay within a specified amount of time.

Predecessor and successor information:

The information especially Node-ID and address about peer's predecessor and successor must be kept in order to maintain the overlay circle. A peer will communicate with its predecessor and successor with keep-alive message (defined in Update message) every several seconds. Of course, this structure can be added as mark-ups or identifiers in the routing table.

Unit leader information:

The information about the unit leader who is responsible for the unit the peer belongs to. A peer can communicate with its unit leader to receive the Update message. Of course, this structure can be added as mark-ups or identifiers in the routing table.

Slice leader information:

The information about the slice leader who is responsible for the slice the peer belongs to. A peer can send event notification message (defined in Update message) to its slice leader to inform events. Of course, this structure can be added as mark-ups or identifiers in the routing table.

The four kinds of information are maintained in all of peers, but there are still some kinds of other data structures should be kept in some special peers like unit leader and slice leader.

Unit boundary identifier:
This identifier does not keep in all the peers, it is stored on the peer who is the boundary of the unit in order to make sure the Update message just being transferred within unit.
Unit boundary peer list:
The unit leader should know the information about the unit's boundary peer in order to know its control area.
Slice leader list:
Every slice leader should know the information about all the slice leaders in the overlay in order to dispatch the event notification messages in time.
Group member list:
Every slice leader should know the information about all the group members in the slice it is responsible in order to exchange group messages with other slice leaders. This list will be used in the two hop lookups algorithm.

Each peer with these data structures can compose a well running overlay topology based on the one hop lookups algorithm.

5. Routing

The routing table of the peer has full information of all the nodes in the overlay. If the peer is not responsible for a resource whose ID is r, it will query the routing table in order to find the first peer whose id is greater than r, which means that the peer should be responsible for this resource; and then sends a request message to the destination node. If no such node is found, it finds the largest Node-ID in the interval between the peer and Resource-ID r.

In the two hop algorithm, when a peer wants to query the successor of a resource, it sends a lookup request to its chosen peer in the slice containing the key of this resource. The chosen peer then examines its own routing table to identify the successor of the key and forwards the request to that peer. So, one peer can find the resource at most two steps.

6. Joining

The Join message defined in [I-D.ietf-p2psip-base] with a Node-ID has contained enough information to construct a joining request.

The joining process for a joining peer (JP) with Node-ID n is as follows (using the one hop lookups algorithm as an example).

(1) JP MUST connect to its chosen or preconfigured bootstrap node.
(2) JP SHOULD send an Attach request to its admitting peer (AP) for Node-ID n. The "send_update" flag should be used to acquire the routing table and other information for AP.
(3) JP MUST send a Join to AP, the AP sends the response to the Join.
(4) AP MUST do a series of Store requests to JP to store the data that JP will be responsible for.
(5) JP SHOULD send Attach request to its predecessor in order to let it knows the arriving of new peer; the predecessor should update its successor information at once.
(6) The predecessor detects a change in its routing table for example it has a new successor, it MUST send an event notification message which is a kind of Update messages to its slice leader directly. Of course, the Update message can also be sent by the AD.
(7) The slice leader MUST collect all notifications it receives from the peers in his slice and sends Update messages to other slice leaders every several seconds to inform these events.
(8) Other slice leaders MUST dispatch the Update messages they received to all the unit leaders in their respective slices.
(9) A unit leader SHOULD piggyback this information on its Update message to its successor and predecessor.
(10) Other nodes propagate this information from their successors; they SHOULD send it to their successors and vice versa.
(11) Peers at unit boundaries SHOULD NOT send information to their neighboring peers outside their unit.

The Update messages depending on the RELOAD message formats will be defined in the next section. The peer can find its successor by sending a Find request with the Resource-ID equals its Node-ID.

7. Updates

The update message is the primary overlay-specific maintenance message which can be used by the sender to notify other peers the current state of overlay. Every peer in the overlay can maintain and keep the correctess of routing table by sending and receiving update messages.

7.1. Update messages definition

The update message defined in this section will be used by the one hop algorithm, D1HT, 1h-Calot and the two hop algorithm. It is a template for one hop like algorithms.

7.1.1. Update types

```
enum { eventUpdate (0), dataStructureUpdate (1), (255) }
  UpdateType;
```

The UpdateType gives an enumeration of update message which includes eventUpdate and dataStructureUpdate. The eventUpdate message will take a list of event notifications which will have an influence on peer's routing table. The dataStructureUpdate message will take a list of data structure information wich includes the routing table items. This structure allows for unknown option types.

7.1.2. Peer types

```
enum { ordinaryPeer (0), unitLeaderPeer (1),
  sliceLeaderPeer (2), (255) } PeerType;
```

The PeerType gives an enumeration of peer type which can be used by different one hop like algorithms. For example, the concepts of unit leader and slice leader are used in heterogeneous DHTs like the one hop lookups and two hop lookups. And in D1HT and 1h-Calot, all the peers are ordinary.

7.1.3. Event notification types

```
enum { peerJoin (0), peerLeave (1), groupPeerInformation (3), (255) }
EventType;
```

The EventType gives an enumeration of event kinds, the common events include peer joining and peer leaving. Almost all of the algorithms need the two events. The third event type is groupPeerInformation which is used in the two hop algorithm to spread group members' information.

7.1.4. PeerContactItem struct

```
struct {
  Node-ID              peer_id
  IpAddressPort        peer_addr_port
} PeerContactItem;
```

The struct of PeerContactItem is used to construct the information about peers. The Node-ID and IpAddressPort are defined in [I-D.ietf-p2psip-base] which represents the ID of the peer and peer's

IP address and port.

7.1.5. EventNotification struct

```
   struct {
     EventType                                                  event_type;
     PeerType                                                   peer_type;
     uint32
length;

     select (event_type) {

       case peerJoin:
         select (PeerType) {
           case ordinaryPeer: case unitLeaderPeer: case sliceLeaderPeer:
             uint32
slice_number;
             uint32
unit_number;
             PeerContactItem                              joining_peer;
             PeerContactItem                              replaced_peer;
         }

       case peerLeave:
         select (PeerType) {
           case ordinaryPeer: case unitLeaderPeer: case sliceLeaderPeer:
             uint32
slice_number;
             uint32
unit_number;
             PeerContactItem                              leaving_peer;
         }

       case groupPeerInformation:
         select (PeerType) {
           case ordinaryPeer: case unitLeaderPeer: case sliceLeaderPeer:

uint32                                                      group_size;

uint32                                                      slice_number;
             PeerContactItem <0...n>                      group_peer_list;
         }
     }
   } EventNotificationItem;
```

The structure of EventNotificationItem is an item of EventNotification message. The peer who receives this item can refresh its routing table and other important data structures by

analysing the information it takes.

The EventNotificationItem structure contains parameters as follows.

event_type
The type of event the message will take, its value is one of the types defined in the EventType. This structure allows for unknown event types.
peer_type
The type of peer who wants to get this information, its value is one of the types defined in the PeerType. This structure allows for unknown peer types.
slice_number, unit_number
The number of slice and unit the peer who caused this event belongs to. These two parameters are only used in the one hop lookup algorithm and the two hop lookup algorithm.
joining_peer
The contact information of the peer's who is going to be one of the overlay members.
replaced_peer
The contact information of the peer's who is replaced by the new arrival one. This variable is only used in the one hop lookup algorithm and the two hop lookup algorithm when the slice leader or unit leader changes.
leaving_peer
The contact information of the peer's who is going to leave the overlay.
group_size
This variable means the number of peers in one group. This variable is only used in the two hop lookup algorithm.
group_peer_list
The representatives or entry points' contact information for other slices. This list is only used in the two hop lookup algorithm.
length
This variable means the length of the remainder of this message.

```
   typedf EventNotificationItem <0...n>
EventNotificationItemList;

   struct {

uint32
uptime;
     PeerContactItem                                          detect_peer;

uint32
D1HT_TTL;

uint32
multicast_range_start;
```

```
uint32
multicast_range_end;

uint32
event_notification_item_counts;
     EventNotificationItemList
event_notification_item_list;
   } EventNotification;
```

The EventNotificaton struct is the most important message body in Updates. This message can be used in the one hop lookup algorithm,

D1HT, 1h-Calot and the two hop lookups algorithm. But different algorithms will use different variables. In the D1HT and 1h-Calot, all the peers are ordinary, so both of them will not use the variables related to unit or slice.

The EventNotification structure contains parameters as follows.

uptime
: The time this peer has been up in seconds.

detect_peer
: This variable means the peer's contact information who detects these events.

D1HT_TTL
: This variable is only used in the D1HT algorithm to be a mark for message spreading. Other algorithms can set it as zero.

multicast_range_start, multicast_range_end
: These two variables mean the multicast range marked by the root of a mulcasting tree. They are only used in the 1h-Calot.

event_notification_item_counts
: This variable means the number of events one update message will take. In the 1h-Calot algorithm, every peer will construct the multicasting tree to spread notifications immediately when it detects the overlay changes. So, the variable can be set as 1 in the 1h-Calot.

event_notification_item_list
: This variable means a list of event notifications. In the one hop lookup algorithm, D1HT and the two hop lookup algorithm, this variable can reduce some network traffic.

7.1.6. DataStructureContent struct

```
   struct {
     PeerType                                                peer_type;
     PeerContactItem                                  predecessor_peer;
     PeerContactItem                                  successor_peer;
     PeerContactItem                                  routing_table_list
<0...n>;
     uint32
length;

     select (peer_type) {
       case ordinaryPeer:
         uint32
unit_number;
         uint32
slice_number;
         PeerContactItem                        unit_leader;
         PeerContactItem                        slice_leader;

       case unitLeaderPeer:
         PeerContactItem                        boundary_peers <2>;

       case sliceLeaderPeer:
         PeerContactItem                        slice_leaders_list <0...n>;
         PeerContactItem                        group_peers_list <0...n>;
     }
   } DataStructureContent
```

The structure DataStructureContent mainly used in the transferring of data structure during the peer joining procedure or receiving a message contains the send_update flag [I-D.ietf-p2psip-base]. The length of the structure changes in different peer type because different peer has different data structure. The ordinary peer only has predecessor, successor, unit leader, slice leader and routing table information. And the unit leader and slice leader should also know some other information.

The DataStructureContent structure contains parameters as follows.

routing_table_list
: This variable represents the peer's routing table which has all of the peers' information in the overlay. In the two hop lookup algorithm, the routing table contains all of the same slice peers's information and other slices' entry peers information.

boundary_peers
: This variable represents the information of peers' who are the boundary of unit. It is used in the one hop lookup algorithm and

the two hop lookup algorithm.

slice_leaders_list
All of the slice leaders' information will be stored in this list, and it can be transferred from one slice leader to another or from the old slice leader to the new one. It is used in the one hop lookup algorithm and the two hop lookup algorithm.

7.1.7. OneHopUpdate message

```
typedef EventNotification<0...n>        EventNotificationList;
typedef DataStructureContent<0...n>     DataStructureContentList;

struct {
  UpdateType           update_type;
  uint32               length;

  select (update_type) {
  case eventUpdate:
    EventNotificationList       event_notification_list;

  case dataStructureUpdate:
    DataStructureContentList    data_structure_list;
  }
} OneHopUpdateReq;
```

This structure is the Update message used in the O (1) protocols. The update message is composed by two kinds of list. Using list can take more information. One of them is the EventNotification structures, and the other is the DataStructureContent structures. All the peers in the overlay can use this update request to inform event notification or transport routing information.

```
struct {
  uint32          update_response;
} OneHopUpdateAns;
```

The structure OneHopUpdateAns is used to be a response to the OneHopUpdateReq message. It is only has a response number to represent the receiver's attitude which may include success, fail and error. This number also can be defined as a type.
update_response
The response of the Update message, different number has different meaning includes success, error and so on.

7.2. Different using strategies

The update message defiend above can cover 4 kinds of O (1) protocols, they are the one hop lookup protocol, D1HT, 1h-Calot and the two hop lookup protocol. Different protocol has different using strategies for the same update message.

7.2.1. One hop lookups

In the one hop lookups, event notifications can be disseminated in a hierarchical architecture. The slice leader collects all the events happened in its responsible slice and spreads this event notification list to all the other slice leaders. The notifications are composed of peer joining and leaving without group information. It also does not use the variable D1HT_TTL which is designed for D1HT specially. Other slice leaders receive the notification list and then send it to the unit leaders it is responsible for. At last, the unit leader spread this list to the ordinary peers via the keep-alive messages.

7.2.2. D1HT

In the D1HT, event can be disseminated by the EDRA (Event Detection and Reporting Algorithm), which is able to notify an event to whe whole system in logartithmic time and yet to have good load-balance properties coupled with very low bandwitdth overhead. This algorithm will use the D1HT_TLL variable defined in the EventNotification structure. In the EDRA, each message will have a Time-To-Live (TTL) counter and will be addressed to its successor. The details of this algorithm can be found in [D1HT]. The update message of D1HT is far different from the one hop lookups algorithm, because all the peers in D1HT are ordinary, so it just use the messages related to the ordinary peers. Other protocols will set the D1HT_TTL as zero or a negative number.

7.2.3. 1h-Calot

The overlay architecture of 1h-Calot is same as D1HT because all the peers are ordinary. In fact, the principles of 1h-Calot are extremely simple: multicast maintains the routing tables; information in the routing table is then used to guide multicast and routing. In the 1h-Calot, a resource is stored on the node whose identifier is the closet to the resource's key in absolute distacnce, regardless of the direction (clockwise or counterclockwise), this is the basic of the routing table maintenance algorithm. A new arrival peer informs other peers of its arrival by multicasting a notification through a tree rooted at the new arrival peer. Every message will take a range variable to tell the next peer its responsible informing boundary. 1h-Calot multicasts each notification through a different tree

expanded just in time according to the current content of the routing tables.

This algorithm will use the multicast_range_start and multicast_range_end variables defined in the EventNotification structure to carry the range information. Beacause the 1h-Calot peer will notify and construct its multicasting tree as soon as the peer detects the overlay changes, so there is only one update message on a multicasting tree which means that the variable event_notification_item_counts will always be one.

7.2.4. Two hop lookups

Although the one hop lookups is a very efficient DHT algorithm, for systems of larger size, the bandwidth requirements of this scheme may become too large for a significant fraction of peers. The two hop lookups schema keeps a fixed fraction of the total routing state on each peer and consumes much less bandwidth, and thus scales to a larger system size.

The overlay architecture of two hop lookups is as same as the one hop lookups. In addition, every slice leader chooses a group of its own peers for each other slice; the group may be chosen randomly or they may be based on proximity metrics. The slice leader sends routing information about one group which can be carried in the group_information_list defined in the EventNotificationItem structure to exactly one other slice leader. The information about the group is then dissemanated to all members of that slice as in the one hop lookups. At las, each peer not only keeps full routing information about peers in its own slice, but also has routing information for the peers in every other slice. The update message dissemanating is still accomplished by the event notification schema used in the one hop lookups.

8. Leaving

The leaving action is also a kind of event, so it can be expressed by the update message. When a peer leaves the overlay peacefully, it needs to send a leaving message to its predecessor or successor which may just include some traditional information like the Node-ID and some security proves like the certificate. And the predecessor or succeor who receives this leaving message will verify its identity and address a update message contains the peer leaving information to its slice leader. The data transfer can also use the update messages, but it sometimes depends on the replication strategies.

If a peer leaves the overlay without any notification, the neighbor

peer (predecessor or successor) must inform this event to the leader by the update messages, and then the backup peer of the leaving one should transfer the backup data to the new one who is charging the leaving peer's namespace. When a unit leader or slice leader leaves this overlay, the overlay should choose a new one to replace the old one.

9. Replication

The replica placement policy of the one hop algorithm can use the way defined in SandStone. It is designed to maximize data reliability and availability, and to maximize network bandwidth utilization. In the SandStone, the subscriber data can be replicated on multiple peers, three on default. We can use a SandStone like replica replacement policy with three replications. The first replica is stored in the primary peer's successor; the second is saved in a peer of different unit but in same slice; and the third is put in a peer from different slice.

Above all, one of the most important issues is the choosing of the replica peer. Maybe we can use a special function whose input is peer's Node-ID to calculate the three replica peers, and of course the root peer should know about the replication method in order to be able to search the data.

10. Fault tolerance

10.1. First hop fail

In the one hop algorithm, if a query fails on its first attempt it does not return an error to an application. Instead, queries can be rerouted with the RELOAD message RouteQueryAns. We can define the RouteQueryAns as follows.

```
struct {
   uint32              response_state;
   PeerContactItem     destination_peer;
} RouteQueryAns;
```

The structure RouteQueryAns is composed by a state number and a Node-ID. The state number tells the query state and the PeerContactItem gives a peer's information that is closer to the destination. And in most of cases, two hops are enough to locate one peer or resource. If the two hops are still wrong, then we can know the lookup procedure is fail.

response_state
The response mark of the RouteQueryReq message's; and it may include success, failure and errors.
destination_peer
If the query is failed, the application can send RouteQueryReq message to this peer again for finding resources.

If a lookup query from peer A to peer B because the routing table is outdate or the peer B has left, peer A can retry the query by sending the RouteQueryReq to the peer C who is the successor of peer B. And at that time joining a peer D to be peer B's new successor and peer C's predecessor, then peer C can reply RouteQueryAns message with the redirect_peer of peer D's, and peer A can contact it in this routing step.

10.2. Leaders fail

The most important issues are how to keep the correct functioning of unit leaders and slice leaders; we need to recover from their failure. When a slice or unit leader fails, its successor soon detects the failure and becomes the new leader; the successor of a failed slice leader will communicate with its unit leaders and other slice leaders to recover information about the missed events.

11. Security Considerations

There is no specific security consideration associated with this draft.

12. IANA Considerations

There are no IANA considerations associated to this memo.

13. Acknowledgements

14. References

14.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[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.

14.2. Informative References

[RFC3174] Eastlake, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, September 2001.

[One-Hop-Lookups] Gupta, A., Liskov, B., and R. Rodrigues, "One Hop Lookups for Peer-to-Peer Overlays", June 2003.

[SandStone] Shi, G., Chen, J., Gong, H., Fan, L., Xue, H., Lu, Q., and L. Liang, "SandStone: A DHT based Carrier Grade Distributed Storage System", September 2009.

[TwoHopLookups] Gupta, A., Liskov, B., and R. Rodrigues, "Efficient Routing for Peer-to-Peer Overlays".

[D1HT] Monnerat, L. and C. Amorim, "D1HT: A Distributed One Hop Hash Table".

[1h-Calot] Tang, C., Buco, M., Chang, R., Dwarkadas, S., Luan, L., Edward, E., and C. Ward, "On the Tradeoff among Capacity, Routing Hops, and Being Peer-to-Peer in the Design of Structured Overlay Networks".

[SingleHopDHT] Monnerat, L. and C. Amorim, "Peer-to-Peer Single Hop Distributed Hash Tables".

Authors' Addresses

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

Email: Penjin@chinamobile.com

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

Email: Lelifeng@chinamobile.com

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

Email: fengkai_sunny@139.com