P2PSIP Working Group
Internet-Draft
Intended status: Informational
Expires: January 13, 2014

D. Bryan
St. Edwards University
P. Matthews
Alcatel-Lucent
E. Shim
Samsung Electronics Co., Ltd.
D. Willis
Softarmor Systems
S. Dawkins
Huawei (USA)
July 12, 2013

Concepts and Terminology for Peer to Peer SIP
draft-ietf-p2psip-concepts-05

Abstract

This document defines concepts and terminology for the use of the Session Initiation Protocol in a peer-to-peer environment where the traditional proxy-registrar and message routing functions are replaced by a distributed mechanism. These mechanisms may be implemented using a distributed hash table or other distributed data mechanism with similar external properties. This document includes a high-level view of the functional relationships between the network elements defined herein, a conceptual model of operations, and an outline of the related problems addressed by the P2PSIP working group and the RELOAD protocol ([I-D.ietf-p2psip-base], [I-D.ietf-p2psip-sip]) defined by the working group.

Table of Contents

1. Editor’s Notes and Changes To This Version

This version of the draft represents a minor revision of version -04 and is intended to restart conversation on this draft in the group, to identify open issues, address them, and complete work on the document.

Version -03 represented a substantial revision from the previous version. Until -02, this work was tracking open questions and being used to help reach consensus on a draft. With the selection of RELOAD as the protocol for this WG, the focus of the group turned to completing the RELOAD drafts, and the WG directed the editors to update the document to reflect the decisions made in RELOAD upon completion.

Please see Section 7 for the list of major open issues.

2. Background

One of the fundamental problems in multimedia communication between Internet nodes is discovering the host at which a given user can be reached. In the Session Initiation Protocol (SIP) [RFC3261] this problem is expressed as the problem of mapping an Address of Record (AoR) for a user into one or more Contact URIs [RFC3986]. The AoR is a name for the user that is independent of the host or hosts where the user can be contacted, while a Contact URI indicates the host where the user can be contacted.

In the common SIP-using architectures that we refer to as "Conventional SIP" or "Client/Server SIP", there is a relatively fixed hierarchy of SIP routing proxies and SIP user agents. To deliver a SIP INVITE to the host or hosts at which the user can be contacted, a SIP UA follows the procedures specified in [RFC3263] to determine the IP address of a SIP proxy, and then sends the INVITE to that proxy. The proxy will then, in turn, deliver the SIP INVITE to the hosts where the user can be contacted.

This document gives a high-level description of an alternative solution to this problem. In this alternative solution, the relatively fixed hierarchy of Client/Server SIP is replaced by a peer-to-peer overlay network. In this peer-to-peer overlay network, the various AoR to Contact URI mappings are not centralized at proxy/registrar nodes but are instead distributed amongst the peers in the overlay.

The details of this alternative solution are specified by the RELOAD protocol. The RELOAD base draft [I-D.ietf-p2psip-base] defines a

mechanism to distribute using a Distributed Hash Table (DHT) and specifies the wire protocol, security, and authentication mechanisms needed to convey this information. This DHT protocol was designed specifically with the purpose of enabling a distributed SIP registrar in mind. While designing the protocol other applications were considered, and when possible design decisions were made that allow RELOAD to be used in other instances where a DHT is desirable, but only when making such decisions did not add undue complexity to the RELOAD protocol. The RELOAD sip draft [I-D.ietf-p2psip-sip] specifies how RELOAD is used with the SIP protocol to enable a distributed, server-less SIP solution.

3. High-Level Description

A P2PSIP Overlay is a collection of nodes organized in a peer-to-peer fashion for the purpose of enabling real-time communication using the Session Initiation Protocol (SIP). Collectively, the nodes in the overlay provide a distributed mechanism for mapping names to overlay locations. This provides for the mapping of Addresses of Record (AoRs) to Contact URIs, thereby providing the "location server" function of [RFC3261]. An Overlay also provides a transport function by which SIP messages can be transported between any two nodes in the overlay.

A P2PSIP Overlay consists of one or more nodes called Peers. The nodes in the overlay collectively run a distributed database algorithm. This distributed database algorithm allows data to be stored on nodes and retrieved in an efficient manner. It may also ensure that a copy of a data item is stored on more than one node, so that the loss of a node does not result in the loss of the data item to the overlay.

One use of this distributed database is to store the information required to provide the mapping between AoRs and Contact URIs for the distributed location function. This provides a location function within each overlay that is an alternative to the location functions described in [RFC3263]. However, the model of [RFC3263] is used between overlays.

3.1. Services

The nature of peer-to-peer computing is that each peer offers services to other peers to allow the overlay to collectively provide larger functions. In P2PSIP, peers offer both distributed storage and distributed message routing services, allowing these functions to be implemented across the overlay. Additionally, the RELOAD protocol offers a simplistic discovery mechanism specific to the TURN

[RFC5766] protocol used for NAT traversal. Individual peers may also offer other services as an enhancement to P2PSIP functionality (for example to support voicemail) or to support other applications beyond SIP. To support these additional services, peers may need to store additional information in the overlay.
[I-D.ietf-p2psip-service-discovery] describes the mechanism used in P2PSIP for resource discovery.

3.2. Clients

An overlay may or may not also include one or more nodes called clients. Clients are supported in the RELOAD protocol as peers that have not joined the overlay, and therefore do not route messages or store information. Clients access the services of the RELOAD protocol by connecting to a peer which performs operations on the behalf of the client. Note that in RELOAD there is no distinct client protocol. Instead, a client connects using the same protocol, but never joins the overlay as a peer. For more information, see [I-D.ietf-p2psip-base].

Note that in the context of P2PSIP, there is an additional entity that is sometimes referred to as a client. A special peer may be a member of the in the P2PSIP overlay and may present the functionality of one or all of a SIP registrar, proxy or redirect server to conventional SIP devices (SIP clients). In this way, existing, non-modified SIP clients may connect to the network. These unmodified SIP devices do not speak the RELOAD protocol, and this is a distinct concept from the notion of client discussed in the previous paragraph.

3.3. Relationship Between P2PSIP and RELOAD

The RELOAD protocol defined by the P2PSIP working group implements a DHT primarily for use by server-less, peer-to-peer SIP deployments. However, the RELOAD protocol could be used for other applications as well. As such, a "P2PSIP" deployment is generally assumed to be a use of RELOAD to implement distributed SIP, but it is possible that RELOAD is used as a mechanism to distribute other applications, completely unrelated to SIP.

3.4. Relationship Between P2PSIP and SIP

Since P2PSIP is about peer-to-peer networks for real-time communication, it is expected that most peers and clients will be coupled with SIP entities (although RELOAD may be used for other applications than P2PSIP). For example, one peer might be coupled with a SIP UA, another might be coupled with a SIP proxy, while a third might be coupled with a SIP-to-PSTN gateway. For such nodes,

the peer or client portion of the node is logically distinct from the SIP entity portion. However, there is no hard requirement that every P2PSIP node (peer or client) be coupled to a SIP entity. As an example, additional peers could be placed in the overlay to provide additional storage or redundancy for the RELOAD overlay, but might not have any direct SIP capabilities.

3.5. Relationship Between P2PSIP and Other AoR Dereferencing Approaches

OPEN ISSUE: Many of the "decisions" made have been moved out of the main document. This one, however, seems to point out a difference. Should this section be moved or removed?

As noted above, the fundamental task of P2PSIP is turning an AoR into a Contact. This task might be approached using zeroconf techniques such as multicast DNS and DNS Service Discovery (as in Apple’s Bonjour protocol), link-local multicast name resolution [RFC4795], and dynamic DNS [RFC2136].

These alternatives were discussed in the P2PSIP Working Group, and not pursued as a general solution for a number of reasons related to scalability, the ability to work in a disconnected state, partition recovery, and so on. However, there does seem to be some continuing interest in the possibility of using DNS-SD and mDNS for bootstrapping of P2PSIP overlays.

3.6. NAT Issues

Network Address Translators (NATs) are impediments to establishing and maintaining peer-to-peer networks, since NATs hinder direct communication between nodes. Some peer-to-peer network architectures avoid this problem by insisting that all nodes exist in the same address space. However, RELOAD provides capabilities that allow nodes to be located in multiple address spaces interconnected by NATs, to allow RELOAD messages to traverse NATs, and to assist in transmitting application-level messages (for example SIP messages) across NATs.

4. Reference Model

The following diagram shows a P2PSIP Overlay consisting of a number of Peers, one Client, and an ordinary SIP UA. It illustrates a typical P2PSIP overlay but does not limit other compositions or variations; for example, Proxy Peer P might also talk to a ordinary SIP proxy as well. The figure is not intended to cover all possible architecture variations, but simply to show a deployment with many common P2PSIP elements.

```
                                                  --->PSTN
     +------+    N     +------+     +---------+  /
     |      |    A     |      |     | Gateway |-/
     |  UA  |####T#####|  UA  |#####|   Peer  |########
     | Peer |    N     | Peer |     |    G    |       #   RELOAD
     |  E   |    A     |  F   |     +---------+       #   P2PSIP
     |      |    T     |      |                       #   Protocol
     +------+    N     +------+                       #    |
        #        A                                     #    |
      NATNATNATNAT                                     #    |
        #                                              #    |   \__/
      NATNATNATNAT                                +-------+  v   /  \
        #        N                                  |       |#####/ UA \
     +------+    A       P2PSIP Overlay             | Peer  |    /Client\
     |      |    T                                  |   Q   |    |___C__|
     |  UA  |    N                                  |       |
     | Peer |    A                                  +-------+
     |  D   |    T                                        #
     |      |    N                                        #
     +------+    A                                        # RELOAD
        #        T                                        # P2PSIP
        #        N    +-------+        +-------+      # Protocol
        #        A    |       |        |       |      #
        #########T####| Proxy |########| Redir |#######
                 N    | Peer  |        | Peer  |
                 A    |   P   |        |   R   |
                 T    +-------+        +-------+
                        |                 /
                        | SIP            /
                  \__/  /               /
                   /\  / _______________/ SIP
                  /  \/ /
                 / UA \/
                /______\
                SIP UA A
```

Figure: P2PSIP Overlay Reference Model

Here, the large perimeter depicted by "#" represents a stylized view of the Overlay (the actual connections could be a mesh, a ring, or some other structure). Around the periphery of the Overlay rectangle, we have a number of Peers. Each peer is labeled with its coupled SIP entity -- for example, "Proxy Peer P" means that peer P which is coupled with a SIP proxy. In some cases, a peer or client might be coupled with two or more SIP entities. In this diagram we have a PSTN gateway coupled with peer "G", three peers ("D", "E" and "F") which are each coupled with a UA, a peer "P" which is coupled

with a SIP proxy, an ordinary peer "Q" with no SIP capabilities, and one peer "R" which is coupled with a SIP Redirector. Note that because these are all Peers, each is responsible for storing Resource Records and transporting messages around the Overlay.

To the left, two of the peers ("D" and "E") are behind network address translators (NATs). These peers are included in the P2PSIP overlay and thus participate in storing resource records and routing messages, despite being behind the NATs.

On the right side, we have a client "C", which uses the RELOAD Protocol to communicate with Proxy Peer "Q". The Client "C" uses RELOAD to obtain information from the overlay, but has not inserted itself into the overlay, and therefore does not participate in routing messages or storing information.

Below the Overlay, we have a conventional SIP UA "A" which is not part of the Overlay, either directly as a peer or indirectly as a client. It does not speak the RELOAD P2PSIP protocol, and is not participating in the overlay as either a Peer nor Client. Instead, it uses SIP to interact with the Overlay via an adapter peer or peers which communicate with the overlay using RELOAD.

Both the SIP proxy coupled with peer "P" and the SIP redirector coupled with peer "R" can serve as adapters between ordinary SIP devices and the Overlay. Each accepts standard SIP requests and resolves the next-hop by using the P2PSIP protocol to interact with the routing knowledge of the Overlay, then processes the SIP requests as appropriate (proxying or redirecting towards the next-hop). Note that proxy operation is bidirectional - the proxy may be forwarding a request from an ordinary SIP device to the Overlay, or from the P2PSIP overlay to an ordinary SIP device.

The PSTN Gateway at peer "G" provides a similar sort of adaptation to and from the public switched telephone network (PSTN).

5. Definitions

This section defines a number of concepts that are key to understanding the P2PSIP work.

Overlay Network: An overlay network is a computer network which is built on top of another network. Nodes in the overlay can be thought of as being connected by virtual or logical links, each of which corresponds to a path, perhaps through many physical links, in the underlying network. For example, many peer-to-peer networks are overlay networks because they run on top of the

Internet. Dial-up Internet is an overlay upon the telephone network. <http://en.wikipedia.org/wiki/P2P_overlay>

P2P Network: A peer-to-peer (or P2P) computer network is a network that relies primarily on the computing power and bandwidth of the participants in the network rather than concentrating it in a relatively low number of servers. P2P networks are typically used for connecting nodes via largely ad hoc connections. Such networks are useful for many purposes. Sharing content files (see <http://en.wikipedia.org/wiki/File_sharing>) containing audio, video, data or anything in digital format is very common, and real-time data, such as telephony traffic, is also exchanged using P2P technology. <http://en.wikipedia.org/wiki/Peer-to-peer>. A P2P Network may also be called a "P2P Overlay" or "P2P Overlay Network" or "P2P Network Overlay", since its organization is not at the physical layer, but is instead "on top of" an existing Internet Protocol network.

P2PSIP: A suite of communications protocols related to the Session Initiation Protocol (SIP) [RFC3261] that enable SIP to use peer-to-peer techniques for resolving the targets of SIP requests, providing SIP message transport, and providing other SIP-related functions. At present, these protocols include [I-D.ietf-p2psip-base], [I-D.ietf-p2psip-sip], [I-D.ietf-p2psip-diagnostics], [I-D.ietf-p2psip-service-discovery] and [I-D.ietf-p2psip-self-tuning].

User: A human that interacts with the overlay through SIP UAs located on peers and clients (and perhaps other ways).

The following terms are defined here only within the scope of P2PSIP. These terms may have conflicting definitions in other bodies of literature. Some earlier versions of this document prefixed each term with "P2PSIP" to clarify the term’s scope. This prefixing has been eliminated from the text; however the scoping still applies.

Overlay Name: A human-friendly name that identifies a specific P2PSIP Overlay. This is in the format of (a portion of) a URI, but may or may not have a related record in the DNS.

Peer: A node participating in a P2PSIP Overlay that provides storage and transport services to other nodes in that P2PSIP Overlay. Each Peer has a unique identifier, known as a Peer-ID, within the Overlay. Each Peer may be coupled to one or more SIP entities. Within the Overlay, the peer is capable of performing several different operations, including: joining and leaving the overlay, transporting SIP messages within the overlay, storing information

on behalf of the overlay, putting information into the overlay, and getting information from the overlay.

Node-ID: Information that uniquely identifies each Node within a given Overlay. This value is not human-friendly -- in a DHT approach, this is a numeric value in the hash space. These Node-IDs are completely independent of the identifier of any user of a user agent associated with a peer.

Client: A node participating in a P2PSIP Overlay but that does not store information or forward messages. A client can also be thought of as a peer that has not joined the overlay. Clients can store and retrieve information from the overlay.

User Name: A human-friendly name for a user. This name must be unique within the overlay, but may be unique in a wider scope. User Names are formatted so that they can be used within a URI (likely a SIP URI), perhaps in combination with the Overlay Name.

Service: A capability contributed by a peer to an overlay or to the members of an overlay. Not all peers and clients will offer the same set of services, and P2PSIP provides service discovery mechanisms to locate services.

Service Name: A unique, human-friendly, name for a service.

Resource: Anything about which information can be stored in the overlay. Both Users and Services are examples of Resources.

Resource-ID: A non-human-friendly value that uniquely identifies a resource and which is used as a key for storing and retrieving data about the resource. One way to generate a Resource-ID is by applying a mapping function to some other unique name (e.g., User Name or Service Name) for the resource. The Resource-ID is used by the distributed database algorithm to determine the peer or peers that are responsible for storing the data for the overlay.

Resource Record: A block of data, stored using distributed database mechanism of the Overlay, that includes information relevant to a specific resource. We presume that there may be multiple types of resource records. Some may hold data about Users, and others may hold data about Services, and the working group may define other types. The types, usages, and formats of the records are a question for future study.

Responsible Peer The Peer that is responsible for storing the Resource Record for a Resource. In the literature, the term "Root Peer" is also used for this concept.

Peer Protocol: The protocol spoken between P2PSIP Overlay peers to share information and organize the P2PSIP Overlay Network. In P2PSIP, this is implemented using the RELOAD [I-D.ietf-p2psip-base] protocol.

Client Protocol: The protocol spoken between Clients and Peers. In P2PSIP and RELOAD, this is the same protocol syntactically as the Peer Protocol. The only difference is that Clients are not routing messages or routing information, and have not (or can not) insert themselves into the overlay.

Peer Protocol Connection / P2PSIP Client Protocol Connection: The TLS, DTLS, TCP, UDP or other transport layer protocol connection over which the RELOAD Peer Protocol messages are transported.

Neighbors: The set of P2PSIP Peers that a Peer or Client know of directly and can reach without further lookups.

Joining Peer: A node that is attempting to become a Peer in a particular Overlay.

Bootstrap Peer: A Peer in the Overlay that is the first point of contact for a Joining Peer. It selects the peer that will serve as the Admitting Peer and helps the joining peer contact the admitting peer.

Admitting Peer: A Peer in the Overlay which helps the Joining Peer join the Overlay. The choice of the admitting peer may depend on the joining peer (e.g., depend on the joining peer’s Peer-ID). For example, the admitting peer might be chosen as the peer which is "closest" in the logical structure of the overlay to the future position of the joining peer. The selection of the admitting peer is typically done by the bootstrap peer. It is allowable for the bootstrap peer to select itself as the admitting peer.

Bootstrap Server: A network node used by Joining Peers to locate a Bootstrap Peer. A Bootstrap Server may act as a proxy for messages between the Joining Peer and the Bootstrap Peer. The Bootstrap Server itself is typically a stable host with a DNS name that is somehow communicated (for example, through configuration, specification on a web page, or using DHCP) to peers that want to join the overlay. A Bootstrap Server is NOT required to be a peer or client, though it may be if desired.

Peer Admission: The act of admitting a node (the "Joining Peer") into an Overlay as a Peer. After the admission process is over, the joining peer is a fully-functional peer of the overlay. During the admission process, the joining peer may need to present credentials to prove that it has sufficient authority to join the overlay.

Resource Record Insertion: The act of inserting a P2PSIP Resource Record into the distributed database. Following insertion, the data will be stored at one or more peers. The data can be retrieved or updated using the Resource-ID as a key.

6. Discussion

6.1. The Distributed Database Function

A P2PSIP Overlay functions as a distributed database. The database serves as a way to store information about Resources. A piece of information, called a Resource Record, can be stored by and retrieved from the database using a key associated with the Resource Record called its Resource-ID. Each Resource must have a unique Resource-ID. In addition to uniquely identifying the Resource, the Resource-ID is also used by the distributed database algorithm to determine the peer or peers that store the Resource Record in the overlay.

Users are humans that can use the overlay to do things like making and receiving calls. Information stored in the resource record associated with a user can include things like the full name of the user and the location of the UAs that the user is using (the users SIP AoR). Full details of how this is implemented using RELOAD are provided in [I-D.ietf-p2psip-sip]

Before information about a user can be stored in the overlay, a user needs a User Name. The User Name is a human-friendly identifier that uniquely identifies the user within the overlay. In RELOAD, users are issued certificates, which in the case of centrally signed certificates, identify the User Name as well as a certain number of Resource-IDs where the user may store their information. For more information, see [I-D.ietf-p2psip-base].

The P2PSIP suite of protocols also standardizes information about how to locate services. Services represent actions that a peer (and perhaps a client) can do to benefit other peers and clients in the overlay. Information that might be stored in the resource record associated with a service might include the peers (and perhaps clients) offering the service. Service discovery for P2PSIP is

defined in [I-D.ietf-p2psip-service-discovery].

Each service has a human-friendly Service Name that uniquely identifies the service. Like User Names, the Service Name is not a resource-id, rather the resource-id is derived from the service name using some function defined by the distributed database algorithm used by the overlay.

A class of algorithms known as Distributed Hash Tables <http://en.wikipedia.org/wiki/P2P_overlay> are one way to implement the Distributed Database. The RELOAD protocol is extensible and allows many different DHTs to be implemented, but specifies a mandatory to implement DHT in the form of a modified Chord DHT. For more information, see [Chord]

6.2. Using the Distributed Database Function

While there are a number of ways the distributed database described in the previous section can be used to establish multimedia sessions using SIP, the basic mechanism defined in the RELOAD base draft and SIP usage is summarized below. This is a very simplistic overview. For more detailed information, please see the RELOAD base draft.

Contact information for a user is stored in the resource record for that user. Assume that a user is using a device, here called peer A, which serves as the contact point for this user. The user adds contact information to this resource record, as authorized by the RELOAD certificate mechanism. The resource record itself is stored with peer Z in the network, where peer Z is chosen by the particular distributed database algorithm in use by the overlay.

When the SIP entity coupled with peer B has an INVITE message addressed to this user, it retrieves the resource record from peer Z. It then extracts the contact information for the various peers that are a contact point for the user, including peer A, and uses the overlay to establish a connection to peer A, including any appropriate NAT traversal (the details of which are not shown).

Note that RELOAD is used only to establish the connection. Once the connection is established, messages between the peers are sent using ordinary SIP.

This exchange is illustrated in the following figure. The notation "Store(U@A)" is used to show the distributed database operation of updating the resource record for user U with the contract A, and "Fetch(U)" illustrates the distributed database operation of retrieving the resource record for user U. Note that the messages between the peers A, B and Z may actually travel via intermediate

peers (not shown) as part of the distributed lookup process or so as to traverse intervening NATs.

```
      Peer B           Peer Z           Peer A
      |                    |                  |
      |                    |         Store(U@Y)|
      |                    |<-----------------|
      |                    |Store-Resp(OK)    |
      |                    |----------------->|
      |                    |                  |
      |Fetch(U)            |                  |
      |------------------->|                  |
      |     Fetch-Resp(U@Y)|                  |
      |<-------------------|                  |
      |                    |                  |
       (RELOAD IS USED TO ESTABLISH CONNECTION)
      |                    |                  |
      | SIP INVITE(To:U)   |                  |
      |----------------------------------->|
      |                    |                  |
```

6.3. NAT Traversal

NAT Traversal in P2PSIP using RELOAD treats all peers as equal and establishes a partial mesh of connections between them. Messages from one peer to another are routed along the edges in the mesh of connections until they reach their destination. To make the routing efficient and to avoid the use of standard Internet routing protocols, the partial mesh is organized in a structured manner. If the structure is based on any one of a number of common DHT algorithms, then the maximum number of hops between any two peers is log N, where N is the number of peers in the overlay. Existing connections, along with the ICE NAT traversal techniques [RFC5245], are used to establish new connections between peers, and also to allow the applications running on peers to establish a connection to communicate with one another.

6.4. Locating and Joining an Overlay

Before a peer can attempt to join a P2PSIP overlay, it must first obtain a Node-ID, configuration information, and optionally a set of credentials. The Node-ID is an identifier that will uniquely identify the peer within the overlay, while the credentials show that the peer is allowed to join the overlay.

The P2PSIP WG does not impose a particular mechanism for how the

peer-ID and the credentials are obtained, but the RELOAD base draft does specify the format for the configuration information, and specifies how this information may be obtained, along with credentials and a Node-ID, from an offline enrollment server.

Once the configuration information is obtained, the RELOAD base draft specifies a mechanism whereby a peer may obtain a multicast-bootstrap address in the configuration file, and can broadcast to this address to attempt to locate a bootstrap peer. Additionally, the peer may store previous peers it has seen and attempt to use these as bootstrap peers, or may obtain an address for a bootstrap peer by some other mechanism. For more information, see the RELOAD base draft.

The job of the bootstrap peer is simple: refer the joining peer to a peer (called the "admitting peer") that will help the joining peer join the network. The choice of admitting peer will often depend on the joining node - for example, the admitting peer may be a peer that will become a neighbor of the joining peer in the overlay. It is possible that the bootstrap peer might also serve as the admitting peer.

The admitting peer will help the joining peer learn about other peers in the overlay and establish connections to them as appropriate. The admitting peer and/or the other peers in the overlay will also do whatever else is required to help the joining peer become a fully-functional peer. The details of how this is done will depend on the distributed database algorithm used by the overlay.

At various stages in this process, the joining peer may be asked to present its credentials to show that it is authorized to join the overlay. Similarly, the various peers contacted may be asked to present their credentials so the joining peer can verify that it is really joining the overlay it wants to.

6.5. Clients and Connecting Unmodified SIP Devices

As mentioned above, in RELOAD, from the perspective of the protocol, clients are simply peers that do not store information, do not route messages, and which have not inserted themselves into the overlay. The same protocol is used for the actual message exchanged. Note that while the protocol is the same, the client need not implement all the capabilities of a peer. If, for example, it never routes messages, it will not need to be capable of processing such messages, or understanding a DHT.

For SIP devices, another way to realize this functionality is for a Peer to behave as a [RFC3261] proxy/registrar. SIP devices then use

standard SIP mechanisms to add, update, and remove registrations and to send SIP messages to peers and other clients. The authors here refer to these devices simply as a "SIP UA", not a "P2PSIP Client", to distinguish it from the concept described above.

6.6. Architecture

The architecture adopted by RELOAD to implement P2PSIP is shown below. An application, for example SIP (or another application using RELOAD) uses RELOAD to locate other peers and (optionally) to establish connections to those peers, potentially across NATs. Messages may still be exchanged directly between the peers. The overall block diagram for the architecture is as follows:

```
 _________________________
|                         |
|    SIP, other apps...   |
|       ___________________|
|      |   RELOAD Layer   |
|______|___________________|
|     Transport Layer     |
|_________________________|
```

7. Open Issues

MAJOR OPEN ISSUE: The initial wording in the high-level description about proving AoR to contact mapping reflects a very long and contentious debate about the role of the protocol, and reflected a pretense that this was an overlay only for P2PSIP. That is explicitly not true in base anymore (see last paragraph of introduction) and the language has been very much genericized in base. Should we make this text more abstract and then use AoR->contact mapping as an example of the (original) use? On a related note, see the last paragraph of the Background section -- do we want to reword this?

OPEN ISSUE: Should we include a section that documents previous decisions made, to preserve the historical debate and prevent past issues from being raised in the future, or simply rely on the mailing list to address these concerns?

OPEN ISSUE: Should we include the use cases from draft-bryan-p2psip-app-scenarios-00 (now long expired)? There was some interest in doing so in previous versions, but no conclusion was reached.

8. Informative References

[Chord] Singh, K., Stoica, I., Morris, R., Karger, D., Kaashock, M., Dabek, F., and H. Balakrishman, "Chord: A scalable peer-to-peer lookup protocol for internet applications", IEEE/ACM Transactions on Neworking Volume 11 Issue 1, pp. 17-32, Feb. 2003.

Copy available at http://pdos.csail.mit.edu/chord/papers/paper-ton.pdf

[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-26 (work in progress), February 2013.

[I-D.ietf-p2psip-diagnostics] Song, H., Jiang, X., Even, R., and D. Bryan, "P2P Overlay Diagnostics", draft-ietf-p2psip-diagnostics-11 (work in progress), March 2013.

[I-D.ietf-p2psip-self-tuning] Maenpaa, J. and G. Camarillo, "A Self-tuning Distributed Hash Table (DHT) for REsource LOcation And Discovery (RELOAD)", draft-ietf-p2psip-self-tuning-08 (work in progress), February 2013.

[I-D.ietf-p2psip-service-discovery] Maenpaa, J. and G. Camarillo, "Service Discovery Usage for REsource LOcation And Discovery (RELOAD)", draft-ietf-p2psip-service-discovery-08 (work in progress), February 2013.

[I-D.ietf-p2psip-sip] Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., Schulzrinne, H., and T. Schmidt, "A SIP Usage for RELOAD", draft-ietf-p2psip-sip-09 (work in progress), February 2013.

[RFC2136] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 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.

[RFC3263] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, June 2002.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4795] Aboba, B., Thaler, D., and L. Esibov, "Link-local Multicast Name Resolution (LLMNR)", RFC 4795, January 2007.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

Authors' Addresses

David A. Bryan
St. Edwards University
Austin, Texas
USA

Email: bryan@ethernot.org

Philip Matthews
Alcatel-Lucent
600 March Road
Ottawa, Ontario K2K 2E6
Canada

Phone: +1 613 784 3139
Email: philip_matthews@magma.ca

Eunsoo Shim
Samsung Electronics Co., Ltd.
San 14, Nongseo-dong, Giheung-gu,
Yongin-si, Gyeonggi-do, 446-712
South Korea

Email: eunsooshim@gmail.com

Dean Willis
Softarmor Systems
3100 Independence Pkwy #311-164
Plano, Texas 75075
USA

Phone: +1 214 504 1987
Email: dean.willis@softarmor.com

Spencer Dawkins
Huawei Technologies (USA)

Phone: +1 214 755 3870
Email: spencerdawkins.ietf@gmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 13, 2014

A. Knauf
T. Schmidt, Ed.
HAW Hamburg
G. Hege
daviko GmbH
M. Waehlisch
link-lab & FU Berlin
July 12, 2013

A RELOAD Usage for Distributed Conference Control (DisCo)
draft-ietf-p2psip-disco-01

Abstract

This document defines a RELOAD Usage for Distributed Conference Control (DisCo) with SIP. DisCo preserves conference addressing through a single SIP URI by splitting its semantic of identifier and locator using a new Kind data structure. Conference members are enabled to select conference controllers based on proximity awareness and to recover from failures of individual resource instances. DisCo proposes call delegation to balance the load at focus peers.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes a RELOAD Usage for distributed conference control (DisCo) in a tightly coupled model with SIP [RFC3261]. The Usage provides self-organizing and scalable signaling that allows RELOAD peers, clients and plain SIP user agents to participate in a managed P2P conference. DisCo defines the following functions:

- o A SIP protocol scheme for distributed conference control
- o RELOAD Usage and definition of conferencing Kind
- o Mechanisms for conference synchronization and call delegation
- o Mechanism for proximity-aware routing within a conference
- o An XML event package for distributed conferences

In this document, the term distributed conferencing refers to a multiparty conversation in a tightly coupled model in which the point of control (i.e., the focus) is identified by a unique URI, but the focus service is located at many independent entities. Multiple SIP [RFC3261] user agents uniformly control and manage a multiparty session. This document defines a new Usage for RELOAD, including an additional Kind code point with a corresponding data structure that complies with the demands for distributed conferences. The 'DisCo' data structure stores the mapping of a single conference URI to multiple conference controllers and thereby separates the conference identifier from focus instantiations.

Authorized controllers of a conference are permitted to register their mapping in the DisCo data structure autonomously. Thus, the DisCo data structure represents a co-managed Resource in RELOAD. To provide trusted and secure access to a co-managed Resource, this document uses the definitions for Shared Resources (ShaRe) [I-D.knauf-p2psip-share].

Delay and jitter are critical issues in multimedia communications. The proposed conferencing scheme supports mechanisms to build an optimized interconnecting graph between conference participants and their responsible conference controllers. Conference members will be enabled to select the closest focus with respect to delay or jitter.

DisCo extends conference control mechanisms to provide a consistent and reliable conferencing environment. Controlling peers maintain a consistent view of the entire conference state. The multiparty system can be re-structured based on call delegation operations.

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].

We use the terminology and definitions from der RELOAD base draft[I-D.ietf-p2psip-base], the peer-to-peer SIP concepts draft [I-D.ietf-p2psip-concepts], the usage for shared resources draft [I-D.knauf-p2psip-share],and the terminology formed by the framework for conferencing with SIP [RFC4353]. Additionally the following terms are used:

Coordinate Value: An opaque string that describes a host’s relative position in the network topology.

Focus peer: A RELOAD peer that provides SIP conferencing functions and implements the Usage for distributed conferencing. It is called ’active’ if already involved in signaling relation to conference participants. Otherwise, if only registered in a distributed conference data structure, it is referred to as a ’potential’ focus peer.

3. Overview of DisCo

3.1. Reference Scenario

The reference scenario for the Distributed Conference Control (DisCo) is shown in Figure 1. Peers are connected via a RELOAD

[I-D.ietf-p2psip-base] instance, in which peers A and B are managing a single multiparty conference. The conference is identified by a unique conference URI, but located at peers A and B fulfilling the role of the focus. The mapping of the conference URI to one or more responsible focus peers is stored in a new RELOAD Resource for distributed conferencing within a data structure denoted as DisCo-Registration. The storing peer SP of the distributed conference resource holds this data.

The focus peers A and B maintain SIP signaling relations to conference participants, which may have different conference protocol capabilities. In this example, peer A is the focus for the RELOAD peer C and the plain SIP user agent E whereas peer B serves as a focus for RELOAD peer D and the RELOAD client F.

RELOAD peers and clients obtain the contact information for the conference from the storing peer. In contrast to this, the user agent E receives the conference URI not by RELOAD mechanisms, but resolves the ID and joins the conference by plain SIP negotiation.

Focus peers maintain a SIP signaling relation among each other used for notification messages that synchronize the conference focus peers' knowledge about the entire conference state. Additionally, focus peers can transfer calls to each other by a call delegation mechanism.

```
              +-------------------+  +------------------+
              |Access Control List|  |DisCo-Registration|
              +-------------------+  +------------------+
                                 \   /
                               +-------+
                               |Storing|
          # # # # # # # # # # | Peer  | # # # # # # # # # # #
         #                    |  SP   |                    #
        #                     +-------+                     #
       #                                                     #
      #                                                       #
     #                                                         #
   +----+                                                    +----+
   |Peer| \               RELOAD Instance                   |Peer|
   | C  |  \                                                | D  |
   +----+   \                                                +----+
      #     SIP                                                #
       #      \                                               #
        #      \                                             #
         #     +-------+                      +-------+     #(
          #    | Focus |                      | Focus |    #  )
           # # | Peer  | # # # # # # # # # # # | Peer  | # #  (
```

```
                |   A   | <===Conf.Events/====> |   B   |       )
                +-------+    Call delegation    +-------+    Overlay
               /                                         \     Comm.
              /                                           \     (
            SIP                                            SIP    )
            /                                                \   (
           /                                                  \   )
      +----------+                                      +--------+
      |User Agent|                                      | Client |
      |    E     |                                      |   F    |
      +----------+                                      +--------+
```

Figure 1: Reference Scenario: Focus peers A,B maintain a distributed conference

3.2. Initiating a Distributed Conference

To create a conference the initiating user agent announces itself as a focus for the conference. It stores its own contact information (Node-ID) as a DisCo-Registration Kind (cf. Figure 2) in the RELOAD overlay. The hashed conference URI is used as the Resource-ID. This Resource will later contain the contact IDs of all potential focus peers including optional topological descriptors.

3.3. Joining a Conference

A RELOAD-aware node (cf. Bob in Figure 2) intending to join an existing conference requests the list of potential focus peers stored in the DisCo-Registration under the conference’s Resource-ID. The node selects any of the focus peers (e.g., Alice) and establishes a connection using AppAttach/ICE [RFC5245]. This transport is then used to send an INVITE to the conference applying the chosen focus as the contact. The selection of the focus peer can optionally be based on proximity information if available.

A conference member proposes itself as a focus for subsequent participants by adding its Node-ID to the DisCo-Registration stored under the conference URI in the RELOAD overlay. The decision whether a peer announces as a focus incorporates bandwidth, power, and other constraints, but details are beyond the scope of this document.

```
Alice                          RELOAD                          Bob
(initiating peer)                                     (joining peer)
-----------------------------------------------------------------------
  |                              |                                |
  |      Alice stores her mapping to register a conference        |
  | Store mapping(ConfURI, Alice) |                               |
  |------------------------------>|                               |
```

```
     |      Bob requests the list of potential focus peers          |
     |                               |        Lookup ConfURI        |
     |                               |<-----------------------------|
     |                               |   Result list of conf. focus |
     |                               |----------------------------->|
     |                               |                              |
     |       Bob establishes transport connection to Alice          |
     |                          AppAttach                           |
     |<-------------------------------------------------------------|
     |                          AppAttach                           |
     |------------------------------------------------------------->|
     |                           INVITE                             |
     |<~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
     |                             OK                               |
     |~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~>|
     |                             ACK                              |
     |<~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
     |                            Media                             |
     |<============================================================>|
     |                               |                              |
     |       Bob stores his mapping to become a focus peer too      |
     |                               |  Store mapping(ConfURI, Bob) |
     |                               |<-----------------------------|
     |                               |                              |
```

Figure 2: DisCo Usage generic Call Flow

3.4. Conference State Synchronization

Each focus of a conference maintains signaling connections to its related participants independently from other conference controllers. This distributed conference design effects that the entire SIP conference state is jointly held by all focus peers. In DisCo, state synchronization is based on a SIP specific event notifications mechanism [RFC3265].

Each focus peer maintains its view of the entire conference state by subscribing to the other focus peers' XML event package for distributed conferences. This is based on the event package for conference state (cf. [RFC4575]). Details are defined in this document in Section 5. Receivers of event notifications update their local conference state document to represent a valid view of current total conference state.

The event notification package for distributed conferences enables focus peers to synchronize the entire conference state. The event package defines additional XML elements and complex types (see Section 8 for more details), which describe the responsibilities of

any focus peer in the conference. By providing a global view each focus peer is enabled to perform additional load balancing operations and enhances the robustness against departures of focus peers.

3.5. Call delegation

Call delegation (see Section 6.1) is a feature used to transfer an incoming participation request to another focus peer. It can be applied to prevent overloading of focus peers. Call delegation is realized through SIP REFER requests, which carry signaling and session description information of the caller to be transferred. This feature is achieved transparently for the transferred user agent by using a source routing mechanism at SIP dialog establishment. Descriptions of overload detection are beyond the scope of this document.

3.6. Resilience

A focus peer can decide to leave the conference or may ungracefully fail. In a traditional conferencing scenario, loss of the conference controller or the media distributor would cause a complete failure of the multiparty conversation. Distributed conferencing uses the redundancy provided by multiple focus peers to reconfigure a current multiparty conversation. Participants that loose their entry point to the conference re-invite themselves via the remaining focus peers or will be re-invited by the latter. This option is based on the conference state and call delegation functions.

3.7. Topology Awareness

DisCo supports the optimization of the conference topology in respect of the underlying network using topological descriptors. An extension for the RELOAD XML configuration document is defined in Section 4.8 to support landmarking approaches. Each peer intending to create or participate in a distributed conference MAY determine a topological descriptor that describes its relative position in the network. Focus peers store these coordinate values in an additional data field in the DisCo-Registration data structure. This enables peers joining the conference to select the closest focus with respect to its coordinate values.

4. RELOAD Usage for Distributed Conference Control

4.1. Shared Resource DisCo-Registration

A distributed conference is a cooperative service of several individual devices that use a common identifier. To enable a mapping from one conference identifier to multiple focus peers, this document defines the new Kind data structure DisCo-Registration. The DisCo Kind uses the definitions for Shared Resources [I-D.knauf-p2psip-share] to allow a jointly maintenance by multiple focus peers. Hence, write permission to a DisCo-registration is shared by the conference creator with all authorized focus peers.

DisCo complies with the following requirements for Shared Resources:

Isolated Data Storage: DisCo uses the dictionary data model. Each dictionary key is the Node-ID of the certificate that will be used to sign the stored data

ResourceNameExtension field: A DisCo-Registration can contain the ResourceNameExtension structure an initial field in the Kind data structure. It contains the conference URI of the registered DisCo-record.

4.2. Kind Data Structure

Each DisCo-Registration data structure stores a mapping of a conference identifier to one or multiple focus peers that cooperatively control the conference. Additionally, each DisCo-Registration provides the coordinate value, which indicates the relative network position of the focus peers.

The data structure uses the RELOAD dictionary type. The dictionary key MUST be the Node-ID of the focus peer that is associated with the dictionary entry. This allows a focus peer to update only its own mapping. The DisCo data structure of type DisCoRegistration is constructed as follows:

```
struct {
  /* This field is optional, see documentation */
  ResourceNameExtension res_name_ext;
  opaque coordinate<0..2^16-1>;
  NodeId node_id;
} DisCoRegistration;
```

The DisCoRegistration structure is composed of the following values:

res_name_ext: This field can contain the conference URI. It meets the requirement for the USER-CHAIN-ACL access policy defined in [I-D.knauf-p2psip-share] to enable variable resource names.

coordinate: This field contains a topological descriptor that indicates the relative position of the peer in the network. To support different algorithms the coordinate field is represented as an opaque string.

node_id: This field contains the Node-ID of the peer storing the DisCoRegistration and is used to contact the peer when utilizing its service as a focus.

4.3. Variable Conference Identifier

DisCo-Registrations can be stored based on a variable Resource Name. However, a conference identifier set by a user MUST follow the requirements for Kinds using variable Resource Names as defined in the ShaRe Usage [I-D.knauf-p2psip-share].

4.4. Conference Creation

The registration of a distributed conference includes the storage of the following two Kinds (see Figure 3).

DisCo-Registration Kind: contains the conference identifier and the optional coordinate value. If used, the coordinate value MAY be determined previously to the conference registration. The Resource Name and hence the Resource-ID is defined by the hash over the desired conference identifier (using the hash algorithm of the overlay).

Access Control List Kind: contains the conference participants that are allowed to register as focus peers for a conference (see [I-D.knauf-p2psip-share]). Its Resource Name/ID is equal to those of the corresponding DisCo-Registration.

Preliminary to storage of DisCo-Registration and Access Control List (ACL) Kinds the conference creator SHOULD perform a RELOAD StatReq to verify that no former conference is present. If neither a DisCo-Registration nor an associated ACL exist, the conference creator stores a DisCo-Registration with a valid conference identifier (see Section 4.3) and an ACL referring to the DisCo-Registration Kind-ID.

If DisCo registrations and ACL Kinds from previous conferences are still existing there are two options. First, if conference creator is aware of the indexes from previous ACL Kinds, it refreshes the root item of this ACL and stores its registration as focus peer as

DisCo-Registration Kind. Second, If the creator is unaware of indexes, it fetches all Access List Kinds to determine the index of the root item.

```
    Alice        Peer1       Overlay       PeerN     Storing Peer
   -------------------------------------------------------------
      |   StatReq Res:Conf-URI                |            |
      |------------>|----------->|----------->|----------->|
      |             StatAns      |            |            |
      |<------------|<-----------|<-----------|<-----------|
      |   StoreReq Res:Conf-URI Kinds:DisCo, Access-List   |
      |------------>|----------->|----------->|----------->|
      |           StoreAns       |            |            |
      |<------------|<-----------|<-----------|<-----------|
      |             |            |            |            |
```

Figure 3: Initial creation of a Distributed Conference

Optionally the conference initiator (or any active focus) MAY store an additional RELOAD SIP-Registration in the overlay [I-D.ietf-p2psip-sip] or even at a standard SIP registrar [RFC3261] under a URI for which it has write permission. This allows DisCo-unaware or even legacy SIP user agents to participate in the conference. Those registrations SHOULD always point to a currently active focus, who is prepared to accept legacy user agents. The user agent who initially performed the registrations is responsible for updating them appropriately. How authorization has been obtained to perform those registration is out of scope of this document.

The lifetime of a distributed conference is not necessarily limited by the participation time of its creator. As long as the root item of an Access Control List to a DisCo-Registration is not expired, Authorized Peers are allowed to further provide a conferencing service and even store new focus registrations.

4.5. Advertising Focus Ability

All participants of a distributed conference MAY potentially become a focus peer for their conference. This depends on its capacities such as sufficient processing power (CPU, Memory) for the desired media type, the quality of the network connectivity, and the conference policy. Information about network connectivity with respect to NAT or restricted firewalls can be obtained via ICE [RFC5245] connectivity checks. If a peer is behind a NAT, it SHOULD allow for incoming connections via AppAttach/ICE. Peers that can only accept connections with the support of TURN should not act as a focus.

Nevertheless, under special circumstances it might be reasonable to locate a focus peer behind such a NAT (e.g., within a companies network infrastructure).

If a participant is a candidate to become a focus for the conference, it stores its Node-ID and optional coordinate value into the DisCo data structure. An entry in the corresponding ACL [I-D.knauf-p2psip-share] must be present to allow this peer to write the DisCo-Registration resource. By storing the mapping into the data structure a participant becomes a potential focus.

4.6. Determining Coordinates

Each RELOAD peer within the context of a distributed conference MAY be aware of its relative position in the network topology. To constuct a topology-aware conference, the DisCo Usage provides the coordinate value within the DisCo-Registration data structure. Focus peers store their relative network position together with the announcement as conference focus. Joining peers that are able to determine their coordinates may then select a focus peer whose relative position is in its vicinity (see Section 4.7).

Some algorithms determine topology information by measuring Round-Trip Times (RTT) towards a set of hosts serving as landmarks (e.g., [landmarks-infocomm02]). To support such algorithms this document describes an extension to the RELOAD XML configuration document that allows to configure the set of landmark hosts peer must use for position estimation (see Section 4.8). Once a focus peer has registered its mapping in the DisCo data structure, it also stores the according coordinates in the same mapping. These <Node-ID,coordinates> vectors are used by peers joining the conference to select the focus peer that is relatively closest to itself.

Because topology-awareness can be obtained by many different approaches a concrete algorithms is out of scope of this document.

4.7. Proximity-aware Conference Participation

The participation procedure to a distributed conference is composed of three operation.

1. Resolution of the conference identifier.

2. Establishment of of transport connection.

3. SIP signaling to join a conference.

Figure 4 and the following specifications give a more detailed view on the joining procedure.

```
     Bob      Peer1     Overlay     PeerN    Storing Peer  Alice
  ---------------------------------------------------------------
     |   StatReq Res:Conf-URI         |          |          |
     |--------->|--------->|--------->|--------->|          |
     |          |StatAns   |          |          |          |
     |<---------|<---------|<---------|<---------|          |
     |   FetchReq Res:Conf-URI Kind:DisCo,ACL    |          |
     |--------->|--------->|--------->|--------->|          |
     |          |FetchAns  |          |          |          |
     |<---------|<---------|<---------|<---------|          |
     |          |          |          |          |          |
     |  Bob calculates Alice as closest Focus    |          |
     |          |          |          |          |          |
     |          |AppAttach application:5060      |          |
     |--------->|--------->|--------->|--------->|--------->|
     |          |AppAttach application:5060      |          |
     |<---------|<---------|<---------|<---------|<---------|
     |          |          |          |          |          |
     |<-------------------ICE Checks----------------------->|
     |          |          |          |          |          |
     |          |         INVITE sip:Alice       |          |
     |~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~>|
     |          |         200 OK      |          |          |
     |<~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~|
     |          |         ACK         |          |          |
     |~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~>|
     |          |          |          |          |          |
```

Figure 4: Participation of a Distributed Conference

1. The joining peer MAY determine its own coordinate value (if used).

2. The joining peer sends a StatReq message to obtain all indexes of the Access Control List (ACL) Kinds stored.

3. The joining peer sends a FetchReq message for the DisCo and ACL Kind to the Resource-ID of the conference URI. The FetchReq SHOULD NOT include any specific dictionary keys, but SHOULD fetch for those array ranges previously determined the StatReq. With the ACL items, the joining peer is able to verify whether DisCo-Registrations are stored by authorized focus peers (see [I-D.knauf-p2psip-share]).

4. Using the retrieved coordinates values of the DisCo-Registrations, the joining peer MAY calculate which of than opaque <0..2^16-1> initial field in the Kind data structure focus peers is the relatively closest to itself.

5. The joining peer then establishes a transport using RELOADs AppAttach, respectively, ICE procedures to create a transport to the chosen focus peer.

6. Finally, the established transport is used to create a SIP dialog from the joining peer to the focus peers.

The SIP INVITE request MAY contain the joining peer's topological descriptor as a URI-parameter called 'coord' in the contact-header in base64 encoded form [RFC4648]. An example contact URI is "sip:alice@example.com;coord=PEknbSBhIHRvcG9sb2dpY2FsIGRlc2NyaXB0b3I+". When the called focus is full and needs to delegate the call it uses the contents of the 'coord'-parameter. It determines the next available focus closest to the calling peer (Section 4.6) using the received descriptor and the other focuses' descriptors from the conference state synchronization document and delegates the call accordingly (see Section 6.1).

A conference focus MAY allow the joining peer to also become a focus (depending on the conference policy see Section 6.2). Therefore, it stores a new ACL Kind that delegates write permission for the DisCo-Registration to the new participant. It sets the 'user_name' field in the ACL Kind to its own user name and sets the 'to_user' field to the user name of the joining peer. If no other conference policy is defined, the focus peer MAY set the allow_delegation flag to true. For further details about the trust delegation using the ACL Kind see [I-D.knauf-p2psip-share].

4.8. Configuration Document Extension

This section defines an additional parameter for the <configuration> element that extends the RELOAD XML configuration document. The proposed <landmarks> element allows RELOAD provider to publish a set of accessible and reliable hosts that SHOULD be used if RELOAD peers use landmarking algorithms to determine relative position in the network topology.

The <landmarks> element serves as container for the <landmark-host> sub-elements, each representing a single host that serves as a landmark. The <landmark-host> uses the following attributes:

address: The IP address (IPv4 or IPv6) of the landmark host.

port: The port on which the landmark host responds for distance estimation.

More than one landmark hosts SHOULD be present in the configuration document.

5. Conference State Synchronization

The global knowledge about signaling and media relations among the conference participants and focus peers defines the global state of a distributed conference. It is composed of the union of every local state at the focus peers. To enable focus peers to provide conference control operations that modify and/or require the global state of a conference, this document defines a mechanism for inter-focus state synchronization. It is based on mutual subscriptions for an Event Package for Distributed Conferences and allows to preserve a coherent knowledge of the global conference state. This XML based event package named 'distributed-conference' MUST be supported by each RELOAD peer that is registered with a DisCo-Registration. Participants of a distributed conference MAY support it. To provide backward compatibility to conference members that do not support the distributed-conference event package, this document defines a translation to the Event Package for Conference State [RFC4575].

5.1. Event Package Overview

The 'distributed-conference' event package is designed to convey information about roles and relations of the conference participants. Conference controllers obtain the global state of the conference and use this information for load balancing or conference restructuring mechanisms in case of a focus failure. Figure Figure 5 gives a general overview of the document hierarchy.

```
distributed-conference
  |
  |-- version-vector
  |    |-- version
  |    |-- version
  |
  |-- conference-description
  |
  |-- focus
  |    |-- focus-state
  |    |    |-- user-count
  |    |    |-- coordinate
  |    |    |-- maximum-user-count
  |    |    |-- active
  |    |    |-- locked
```

```
                      |    |    |-- conf-uris
                      |    |    |-- available-media
                      |    |
                      |    |-- users
                      |    |    |-- user
                      |    |    |    |-- endpoint
                      |    |    |    |    |-- media
                      |    |    |    |    |-- call-info
                      |    |
                      |    |-- relations
                      |    |    |-- relation
                      |-- focus
                      |    |-- ...
```

Figure 5: Overview of the event package for distributed conferences

The document structure is designed to allow concurrent change events at several focus peers. To prevent race conditions each focus peer has exclusive writing permission to the 'focus' sub element that describes itself. It is achieved by a unique mapping from a focus peer to its XML element using the 'Element Keys' mechanisms for partial notification [RFC4575](sections 4.4-5.). A focus peer is only allowed to update or change that <focus> sub element, whose 'entity' Element Key contains its RELOAD user name. This restriction also applies to the child elements of the 'version-vector' element. Each 'version' number belongs to a specific focus peer maintaining the version number.

The local state of a focus peer is described within a 'focus' element. It provides general information about a focus peer and its signaling and media relations to participants and focus peers. The Conference participants are aggregated within 'users' elements, respectively, 'user' sub elements.

General information about the conference as a whole, is provided within a 'conference-description' element. In contrast to the 'focus' and 'version-vector' elements, conference description is not meant for concurrent updating. Instead, it provides static conference descriptions that rarely change during a multiparty session.

More detailed descriptions about the event package and its elements are provided in the following sections. The complete XML schema definition is given in Section 8.

5.2. <distributed-conference>

The <distributed-conference> element is the root of a distributed conference XML document. It uses the following attributes:

entity: This attribute contains the conference URI that identifies the distributed conference. A SIP SUBSCRIBE request addressed to this URI initiates an subscribe/notify relation between participants and their related focus peer.

state: This attribute indicates whether the content of a distributed conference document is a 'full', 'partial' or 'deleted' information. It is in accordance with [RFC4575] to enable the partial notification mechanism.

The <distributed-conference> child elements are <vector-version>, <conference-description> and the <focus> elements. An event notification of state 'full' MUST include all these elements. An event notification of state 'partial' MUST contain at least <version-vector> and its sub elements.

5.3. <version-vector>/<version>

The Event Package for Distributed Conferences uses the <version-vector> and its <version> sub elements to enable a coherent versioning scheme. It is based on vector clocks as defined in [timestamps-acsc88]. Each <version> element contains a unsigned integer that describes the state of a specific focus peer and contains the following attributes:

entity: This attribute contains the user name of the focus peer whose local version number is described by this element.

node-id: This attribute contains the Node-ID of the focus peer.

Whenever the local status of a focus peer changes (e.g. a new participant arrived) the version number of the corresponding <version> element MUST be incremented by one. Each change in the local state also triggers a new event notification containing the entire <version-vector> and the changes contained in a <focus> element.

The recipient of a change event needs to update it local XML document. If a received <version> number is higher that the local, it updates the <version> element and its associated <focus> element with the retrieved elements. All other elements remain constant.

If the length or contents of the retrieved <version-vector> is different to the local copy it indicates a incoherent knowledge about the entire conference state. If the retrieved <version-vector>

contains any unknown focus peers and any local version numbers for the known focus peers is lower, the receiver SHOULD request a 'full' XML notification.

If any local <version> number is retarded more than one, the receiver SHOULD request a 'full' event notification from the sender. The full state notification updates all <focus> elements whose corresponding <version> element is out of date.

5.4. <conference-description>

The <conference-description> element provides general information and links to auxiliary services for the conference. The following sub elements provide human-readable text descriptions about the conference:

<display-text>: Contains a short textual description about the conference

<subject>: Contains the subject of the conference

<free-text>: Contains a longer textual description about the conference

<keywords>: Contains a list of keywords that match the conference topic. The XML schema definition and semantic is imported from the 'conference-info' event package [RFC4575].

The <service-uris> sub element enables focus peers to advertise auxiliary services for the conference. The XML schema definition and semantic is imported from the 'conference-info' event package [RFC4575].

The <conference-description> element uses the 'state' Element Key to enable the partial notification mechanism.

5.5. <focus>

Each <focus> element describes a focus peer actively controlling the conference. It provides general information about a focus peer (e.g., display-text, languages, etc.), contains conference specific information about the state of a focus peer (user-count, available media types, etc.) and announces signaling and media information about the maintained participants. Additionally, it describes signaling or media relations to further focus peers.

The <focus> element uses the following attributes:

entity: This attribute contains the user name of the RELOAD peer acting as focus peer. It uniquely identifies the focus peer that is allowed to update or change all sub elements of <focus>. All other focus peers SHOULD NOT update or change sub elements of this <focus> element. A SUBSCRIBE request addressed to the user name initiates a conference state synchronization with the focus peer.

Node-ID This attributed contains the Node-ID of the peer acting as conference focus

state: In accordance to [RFC4575], this attribute indicates whether the content of the <focus> element is a 'full', 'partial' or 'deleted' information. A 'partial' notification contains at maximum a single <focus> element.

The following sub elements of <focus> provide general information about a focus peer:

<display-text>: Contains a short text description of the user acting as focus peer.

<associated-aors>: This element contains additional URIs that are associated with this user.

<roles:> This element MAY contain human-readable text descriptions about the roles of the user in the conference.

<languages>: This element contains a list of tokens, each describing a language understood by the user.

The XML schema definition and semantic for <associated-aors>, <roles> and <languages> are imported from the 'conference-info' event package [RFC4575]

Following, a detailed description of the remaining sub elements.

5.5.1. <focus-state>

The <focus-state> element aggregates a set of conference specific information about the RELOAD user acting as focus peer. The following attribute is defined for the <focus-state> element:

status: This attribute indicates whether the content of the <focus-state> element is a 'full', 'partial' or 'deleted' information.

The <focus-state> element has the following sub elements:

<user-count>: This element contains the number of participants that are connected to the conference via this focus peer at a certain moment.

<coordinate> This element contains the coordinate value Section 4.2 of the focus peer

<maximum-user-count>: This number indicates a threshold of participants a focus peer is able to serve. This value might change during a conference, depending on the focus peers current load.

<conf-uris>: This element MAY contain other conference URIs in order to access the conference via different signaling means. The XML schema definition and semantic is imported from [RFC4575].

<available-media>: This element is imports the <conference-media-type> type XML scheme definitions from [RFC4575]. It allows a focus peer to list its available media streams.

<active>: This boolean element indicates whether a focus peer is currently active. Conference participation requests or a call delegation request SHOULD succeed.

<locked>: In contrast to <active>, this element indicates that a focus peer is not willing to accept anymore participation or call delegation request.

5.5.2. <users>/<user>

The <users>, respectively, each <user> element describes a single participant that is maintained by the focus peer described by the parent <focus> element. The <users> element XML schema definition and its semantic is imported from the 'conference-info' event package [RFC4575].

5.5.3. <relations>/<relation>

The <relations> element serves as container for <relation> elements, each describing a specific connection to another focus peer. The parent element <relations> uses the 'state' attribute to enable the partial notification mechanism. For the <relation> element the following attributes are defined:

entity: This attribute contains the user name of the remote focus.

node-id This attribute contains the Node-ID of the remote focus peer.

The content of each <relation> is a comma separated string that describes the tuple <CONNECTION-TYPE:IDENTIFIER>. The CONNECTION-TYPE is a string token describing the type of connection (e.g. media, signaling, etc.) and the IDENTIFIER contains a variable connection identifier. It is a generic method to announce any kind of connection to a remote focus. This specification defines following tuples:

media:<label>: This tuple identifies a single media stream. The 'label' variable contains the SDP "label" attribute. (see [RFC4574]).

sync:<call-id>: This tuple indicates a subscription for the Event Package for Distributed Conferences. The 'call-id' variable contains the call-id of the SIP subscription dialog.

5.6. Distribution of Change Events

Each focus peer in a distributed conference must be able to retrieve all change events from all other focus peers. A simple approach would be to inter-connect each focus with all other focus in a full meshed topology. To avoid a full mesh, this document describes a 'mutual' subscription scheme that constructs a spanning tree topology among the focus peers.

A conference participant that becomes a focus peer MUST send a SIP SUBSCRIBE to request the Event Package for Distributed Conferences to its own focus peer. The subscription request is addressed to user name of the active focus peer. The latter interprets this subscription as a request for conference state synchronization. The corresponding NOTIFY message contains a 'full' distributed-conference state XML document (see section Section Section 5.1).

The subscribed focus peer in turn subscribes the upcoming focus peer for the distributed conference event package. The corresponding NOTIFY message carries a 'partial' conference state XML document. It contains the received <version-vector> including a new <version> element for itself and a new <focus> element that describes its local state (see Section 5.5).

Resulting by this subscription scheme, each focus peer has at least one subscription to obtain updates for the conference state and is a notifier for change events originated itself. In a incrementally increasing conference, the 1st and 2nd focus peer have a mutual subscription for conference change events. A 3rd focus could have a mutual subscription with the 1st focus, a 4th focus to the 2nd focus and so forth. The result is a spanning tree topology among the focus peers in which each focus peer is a possible root for distribution tree for conference change events.

However, the fact that event notifications need to traverse one or more intermediate focus peers until conference-wide delivery, demands a forwarding mechanism for change events. On receiving a change event, a notified focus validates based on the <version-vector> whether the incoming state event is not a duplicate to previews notifications. If its not a duplicate, the received change event triggers a new event notification at the receiver of the change event. It notifies all its subscribers with excepting the sender of the event notification. In this way, the change event will be 'flooded' among the focus peer of a conference.

5.7. Translation to Conference-Info Event Package

The Event Package for Distributed Conferences imports several XML element definitions of the Event Package for Conference State [RFC4575]. This is caused by two reasons. First, the semantic of these elements are fitting the demands to describe the global state of a distributed conference and, second, it facilitates a re-translation to [RFC4575] to enable a backward compatibility to DisCo-unaware clients. Therefore, each focus peer MAY provide a separate [RFC4575] conform event notification service to its connected participants.

The following sections describe the translation to the Event Package for Conference State [RFC4575] by defining translation rules for the root element and its direct sub elements. For a better understanding, the following sections use a notation ci.<ELEMENT> to refer to a sub element of the conference-info element, and disco.<ELEMENT> to refer to an element of the distributed-conference event package.

5.7.1. <conference-info>

The root element of Event Package for Conference State uses the attributes 'entity', 'state' and 'version' and is the counterpart of the <distributed-conference> root element in the DisCo Event Package. The former two attributes 'entity' and 'state' are equal in both root elements and can be seamlessly translated.

According to [RFC4575], the 'version' attribute SHOULD be incremented by one at any time a new notification being sent to a subscriber. Hence, in DisCo the 'version' attributed increments with each change event that originated by focus peer and each reception of a change events of remote focus peer.

5.7.2. <conference-description>

The <conference-description> element exists in both event packages, conference-info and distributed-conference. Thus, the following elements are seamlessly translatable: <keywords>, <display-text>, <subject>, <free-text> and <service-uris>.

The sub elements <conf-uris>, <maximum-user-count> and <available-media> in conference-info have there counterparts below the \focus \focus-state element of the distributed-conference event package. Each describes a local state of a focus peer in the conference. Hence, the intersection of every disco.<conf-uris>, disco.<available-media> and the sum over each disco.<maximum-user-count> element of each disco.<focus> element in distributed-conference, specifies the content of the corresponding conference-info elements.

5.7.3. <host-info>

According to [RFC4575] the ci.<host-info> element contains information about the entity hosting the conference. For participants in a distributed conference, the hosting entity is their focus peer. Thus, the ci.<host-info> element contains information about a focus peer.

5.7.4. <conference-state>

The ci.<conference-state> element allows subscribers obtain information about overall state of a conference. Its sub elements ci .<user-count>, ci.<active> and ci.<locked> are reused as sub elements of \focus\focus-state to describe the local state of a focus peer in a distributed conference. The translation rules from the distributed-conference to the conference-info event package are the following:

<user-count>: The sum over each value of the disco.<user-count> element defines the corresponding ci.<user-count>.

<active>: The boolean ci.<active> element is the logical concatenation over all disco.<active> elements by an OR-operator.

<locked> The boolean ci.<locked> element is the logical concatenation over all disco.<locked> elements by an AND-operator.

5.7.5. <users>/<user>

The distributed-conference event package imports the definitions of the ci.<users> and ci.<user> elements under a parent disco.<focus> element for each focus peer in a conference. Thus, the aggregation over all disco.<users> elements specifies the content of the corresponding ci.<users> element.

5.7.6. <sidebars-by-ref>/<sidebars-by-value>

In accordance to [RFC4575], if a participant is connected to a sidebar, its responsible focus peer creates a new <user> by referencing to the corresponding sidebar conference.

6. Distributed Conference Control with SIP

Distributed conference control with SIP defined in this document refers to multiparty conversation in a tightly coupled model that is controlled by several independent entities. It enables a resilient conferencing service for P2P scenarios and provides mechanisms for load-balancing among the focus peers. This section describes additional control operations for distributed conferences with SIP.

6.1. Call delegation

Distributed conference control enables load-balancing by a mechanism for call delegation. Call delegations are performed by focus peers that are running out of capacities to serve more participants. Incoming participation requests are then transferred to other established focus peer or conference participants that are registered as potential focus peers in the overlay. Call delegations use SIP REFER requests [RFC3515] that contain additional session information and are achieved transparently to the transferred party.

A focus peer initiates a call delegation by sending SIP REFER request containing the URI of the participant in the Refer-To header field. Additionally, the focus peer appends the following parameter to the URI of the participant:

call-id: Contains the call-ID of the initial SIP dialog between the referred participant and the referring focus peer.

sess-id: Contains the ’session identifier’ value of the original SDP ’o=’ field of the original offer/answer process between referred participant and referring focus peer.

If the recipient accepts the REFER request, it generates a re-INVITE towards the referred party and sets the SIP call-id header and the

SDP 'session-identifier' field in the SDP offer, according to the URI parameter values of the initial REFER request. The From header field and contact header are set to the conference URI with setting the 'isfocus' tag to contact header. This identifies the peer as a focus to the conference and identifies this re-INVITE as a request of the SIP dialog between the party and the conference. To ensure that further signaling messages will be routed correctly, the new focus adds a Record-Route header field that contains its contact information (URI, IP-address,..).

An example call flow for call delegation is shown in Figure 6.

```
Participant    Referring Focus                          Remote Focus
-----------------------------------------------------------------
     |    Dialog    |                                    |
     |<============>|                                    |
     |              |                                    |
     |  Delegating a participant to remote focus         |
     |              |                                    |
     |              |(1) REFER refer-to:<uri>?call-id=123&sess-id=456
     |              |------------------------------------>|
     |              |(2) 200 OK                           |
     |              |<------------------------------------|
     |              |(3) Notify: pending                  |
     |              |<------------------------------------|
     |              |(4) 200 OK                           |
     |              |------------------------------------>|
     |              |                                     |
     |  Remote focus adds RR-header that carries its URI  |
     |              |                                     |
     |  (5) INVITE sip:<uri> record-route:<rem.focus>     |
     |<----------------------------------------------------|
     |  (6) 200 OK  |                                     |
     |---------------------------------------------------->|
     |  (7) ACK     |                                     |
     |<----------------------------------------------------|
     |              |(8) Notify: active                   |
     |              |<------------------------------------|
     |              |(9) 200 OK                           |
     |              |------------------------------------>|
```

Figure 6: Delegating a participant with SIP REFER

Note, subscriptions for the event packages 'distributed-conference' and 'conference-info' are in scope of a specific focus peer and its connected participants. Hence, after a successful call delegation, the referring focus peer SHOULD terminate any subscription to the referred participant. The notifier SHOULD include a reason parameter

"deactivated" to indicate a migration of the subscription as defined in [RFC3265]. The new SUBSCRIBE request by the party MUST be sent via the SIP dialog to the conference.

6.2. Conference Access

A conference policy defines who is allowed to participate in a multimedia conference. In many cases, a group conversation can be an open discussion free to participate, while in other occasions a closed privacy of a multiparty session is demanded. In distributed conferences, it is also an issue which of the conference participants is allowed to become a controller of the multiparty session.

Thus it must be decided whether:

- o A peer is allowed to participate in a conference

- o A peer is allowed to become a focus of the conference

Standard SIP authentication mechanisms can be used to authenticate and accordingly authorize joining participants.

6.3. Media Negotiation and Distribution

This section describes a basic scheme for media negotiation and distribution, which is done in an ad-hoc fashion.

In an established DisCo conference, each participant is attached to one focus (possibly itself), and all focus peers maintain mutual signaling relationships. Each focus peer receives media streams from two groups, its locally attached members, as well as the neighboring focus peers. It has two options when re-distributing media streams to the conference. It either mixes streams from each group and thus reduces the number of media sessions, or propagates the streams in individual media sessions.

The basic media distribution naturally follows the SIP signaling paths. Each focus peer forwards all media streams it receives from the conference (possibly mixed) to all connected peers it is responsible for, and similarly all streams from its peers to its responsible focus. Implementations can choose more sophisticated schemes for media distribution, e.g., some form of overlay multicast, but MUST take measures to prevent loops in media routing.

6.3.1. Offer/Answer

A peer joining a conference negotiates media types and media parameters with its designated focus using the standard SDP offer/

answer protocol [RFC3264]. The focus SHOULD offer all existing media streams that it receives from the conference.

A new participant does not necessarily know about all media streams present in a conference beforehand, and thus some of the media streams might not be included in the initial SDP offer sent by the joining peer. An SDP answer sent by the corresponding focus though cannot offer additional media types that do not match an offer (cf. [RFC3264] Section 6). A joining peer, which is aware of a heterogeneous conference, can invert the offer/answer dialog by omitting an SDP offer in the initial INVITE message. Instead it MAY send an empty INVITE to which the focus replies with an OK, containing the SDP offer. This prevents the need for a second offer/answer-dialog to modify the session. But for compatibility the normal behavior with the INVITE containing the offer MUST be supported.

For new media streams (e.g. those sent by the new participant), the focus SHOULD only offer media types and codecs which are already used in the conference and which will probably be accepted when forwarded to neighboring peers, unless the focus is prepared to do transcoding and/or mixing of the received streams.

6.3.2. New Peers Joining

When a new peer has been attached to a focus, new media streams may be available to the focus that need to be forwarded to the conference. To accomplish this, the new media streams need to be signalled to the other participants. This is commonly done by sending a SIP re-INVITE [RFC4353] for modifying the media sessions, adding the new streams to the SDP. This renegotiation can be costly since it needs to be propagated throughout the entire conference. In addition, distributing all media streams separately to all participants can be quite bandwidth intensive. Both problems can partially be mitigated by focus peers performing mixing of media streams, thus trading bandwidth and signalling overheads for computational load on focus peers.

6.4. Restructuring a Conference

Distributed conference control provides the possibility to delegate calls to remote focus peers. This feature is used to restructure a conference in case of departure of a focus peer. Following, this section presents restructuring schemes for graceful and unexpected leaves of conference focus peers.

6.4.1. On Graceful Leave

In a graceful case, the leaving focus peer (LP) accomplishes the following procedure:

- o LP deletes its mapping in the DisCo-Registration by storing the "non-existing" value as described in the RELOAD base document [I-D.ietf-p2psip-base]. Afterwards, it fetches the lasted version of the DisCo-Registration to obtain all potential focus peers.

- o LP calculates for all its participants the closest focus among all active and potential focus peer using the algorithm described in Section 4.6. LP then delegates all participants to those focus peers.

- o LP then announces its leave by sending a NOTIFY to all its subscribers for the extended conference event package, setting its <focus> state to ’deleted’. Thereafter, it ends its own SIP conference dialog by sending by to its related focus peer.

Since the state synchronization topology in a distributed conference is commonly arranged in a spanning tree, a leave of a focus peer effects a gab in the tree structure. Those focus peers which had the leaving focus peer as their parent, are supposed to reconnect to the synchronization graph by subscribing the parent focus of the leaving peer.

6.4.2. On Unexpected Leave

If an unexpected leave is detected by a participant (e.g. missing signaling and/or media packets) it MUST repeat the joining procedure as described in Section 4.7.

7. DisCo Kind Definition

This section formally defines the DisCo kind.

Name
: DISCO-REGISTRATION

Kind IDs
: The Resource name DISCO-REGISTRATION Kind-ID is the AoR of the conference. The data stored is the DisCoRegistrationData, that contains the Node-ID of a peer acting as a focus for the conference and optionally a coordinates value describing a peer’s relative network position.

Data Model
: The data model for the DISCO-REGISTRATION Kind-ID is dictionary. The dictionary key is the Node-ID of the peer action as focus.

Access Control
USER-CHAIN-ACL

The data stored for the Kind-ID DISCO-REGISTRATION is of type DisCoRegistration. It contains a "coordinates" value, that describes the peers relative network position and the Node-ID of the registered focus peer.

8. XML Schema

The XML schema for the event package for distributed conferences is:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:ci="urn:ietf:params:xml:ns:conference-info"
     xmlns="urn:ietf:params:xml:ns:distributed-conference"
     targetNamespace="urn:ietf:params:xml:ns:distributed-conference"
     elementFormDefault="qualified"
     attributeFormDefault="unqualified">
<!--
    This imports the definitions in conference-info
-->
<xs:import namespace="urn:ietf:params:xml:ns:conference-info"
        schemaLocation="http://www.iana.org/assignments/xml-registry/
                       schema/conference-info.xsd"/>
<xs:import namespace="http://www.w3.org/XML/1998/namespace"
     schemaLocation="http://www.w3.org/2001/03/xml.xsd"/>
  <!--
   A DISTRIBUTED CONFERENCE ELEMENT
-->
<xs:element name="distributed-conference"
            type="distributed-conference-type"/>
  <!--
    DISTRIBUTED CONFERENCE TYPE
-->
  <xs:complexType name="distributed-conference-type">
    <xs:sequence>
      <xs:element name="version-vector"
            type="version-vector-type" minOccurs="1"/>
      <xs:element name="conference-description"
      type="conference-description-type"
      minOccurs="0" maxOccurs="1"/>
      <xs:element name="focus"
      type="focus-type"
      minOccurs="0"
      maxOccurs="unbounded"/>
       <xs:any namespace="##other" processContents="lax"/>
```

```
        </xs:sequence>
        <xs:attribute name="state" type="ci:state-type"/>
        <xs:attribute name="entity" type="xs:anyURI"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:complexType>
    <!--
        VERSION VECTOR TYPE
    -->
      <xs:complexType name="version-vector-type">
        <xs:sequence>
          <xs:element name="version"
                      type="version-type"
                      minOccurs="1"
                      maxOccurs="unbounded"/>
          <xs:any namespace="##other" processContents="lax"/>
        </xs:sequence>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:complexType>
    <!--
        CONFERENCE DESCRIPTION TYPE
    -->
      <xs:complexType name="conference-description-type">
        <xs:sequence>
          <xs:element name="display-text"
                      type="xs:string" minOccurs="0"/>
          <xs:element name="subject" type="xs:string" minOccurs="0"/>
          <xs:element name="free" type="xs:string" minOccurs="0"/>
          <xs:element name="keywords"
                      type="ci:keywords-type" minOccurs="0"/>
          <xs:element name="service-uris"
                      type="ci:uris-type" minOccurs="0"/>
          <xs:any namespace="##other" processContents="lax"/>
        </xs:sequence>
        <xs:attribute name="state" type="ci:state-type"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:complexType>
    <!--
        FOCUS TYPE
    -->
      <xs:complexType name="focus-type">
        <xs:sequence>
          <xs:element name="display-text"
                      type="xs:string" minOccurs="0"/>
          <xs:element name="associated-aors"
                      type="ci:uris-type" minOccurs="0"/>
          <xs:element name="roles"
                      type="ci:user-roles-type" minOccurs="0"/>
          <xs:element name="languages"
```

```
                    type="ci:user-languages-type" minOccurs="0"/>
        <xs:element name="focus-state"
                    type="focus-state-type" minOccurs="0"/>
        <xs:element name="users"
                    type="ci:users-type" minOccurs="0"/>
        <xs:element name="relations"
                    type="relations-type" minOccurs="0"/>
        <xs:any namespace="#other" processContents="lax"/>
      </xs:sequence>
      <xs:attribute name="entity" type="xs:anyURI"/>
      <xs:attribute name="node-id" type="xs:string"/>
      <xs:attribute name="state" type="ci:state-type"/>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
  <!--
      VERSION TYPE
  -->
    <xs:complexType name="version-type">
      <xs:simpleContent>
        <xs:extension base="xs:unsignedInt">
          <xs:attribute name="entity" type="xs:anyURI"/>
          <xs:attribute name="node-id" type="xs:string"/>
          <xs:anyAttribute namespace="##other" processContents="lax"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
    <!--
      FOCUS STATE TYPE
  -->
    <xs:complexType name="focus-state-type">
      <xs:sequence>
        <xs:element name="user-count"
                    type="xs:unsignedInt" minOccurs="0"/>
        <xs:element name="coordinate"
                    type="xs:string" minOccurs="0"/>
        <xs:element name="maximal-user-count"
                    type="xs:unsignedInt" minOccurs="0"/>
        <xs:element name="conf-uris"
                    type="ci:uris-type" minOccurs="0"/>
        <xs:element name="available-media"
                    type="ci:conference-media-type" minOccurs="0"/>
        <xs:element name="active" type="xs:boolean" minOccurs="0"/>
        <xs:element name="locked" type="xs:boolean" minOccurs="0"/>
        <xs:any namespace="##other" processContents="lax"/>
      </xs:sequence>
      <xs:attribute name="state" type="ci:state-type"/>
      <xs:anyAttribute namespace="##other" processContents="lax"/>
    </xs:complexType>
```

```
  <!--
    RELATIONS TYPE
-->
  <xs:complexType name="relations-type">
    <xs:sequence>
      <xs:element name="relation"
                  type="relation-type"
                  minOccurs="0" maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="lax"/>
    </xs:sequence>
    <xs:attribute name="state" type="ci:state-type"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>
  <!--
    RELATION TYPE
-->
  <xs:complexType name="relation-type">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="entity" type="xs:anyURI"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:schema>
```

Figure 7

9. Relax NG Grammar

The grammar for the Landmark configuration document extension is:

```
<!--
    LANDMARKS ELEMENT
-->
parameter &= element landmarks {
    attribute version { xsd:int }
    <!--
        LANDMARK-HOST ELEMENT
    -->
    element landmark-host {
        attribute address { xsd:string },
        attribute port { xsd:int }
    }*
}?
```

Figure 8

10. Security Considerations

10.1. Trust Aspects

TODO

11. IANA Considerations

TODO: register Kind-ID code point at the IANA

12. Acknowledgments

This work was stimulated by fruitful discussions in the P2PSIP working group and SAM research group. We would like to thank all active members for constructive thoughts and feedback. In particular, the authors would like to thank (in alphabetical order) David Bryan, Toerless Eckert, Lothar Grimm, Cullen Jennings, Peter Musgrave, Joerg Ott, Peter Pogrzeba, Brian Rosen, and Jan Seedorf.

13. References

13.1. 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-26 (work in progress), February 2013.

[I-D.knauf-p2psip-share]
Knauf, A., Hege, G., Schmidt, T., and M. Waehlisch, "A Usage for Shared Resources in RELOAD (ShaRe)", draft-knauf-p2psip-share-03 (work in progress), April 2012.

[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.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, June 2002.

[RFC3265] Roach, A., "Session Initiation Protocol (SIP)-Specific Event Notification", RFC 3265, June 2002.

[RFC3515] Sparks, R., "The Session Initiation Protocol (SIP) Refer Method", RFC 3515, April 2003.

[RFC4574] Levin, O. and G. Camarillo, "The Session Description Protocol (SDP) Label Attribute", RFC 4574, August 2006.

[RFC4575] Rosenberg, J., Schulzrinne, H., and O. Levin, "A Session Initiation Protocol (SIP) Event Package for Conference State", RFC 4575, August 2006.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

13.2. Informative References

[I-D.ietf-p2psip-concepts]
Bryan, D., Willis, D., Shim, E., Matthews, P., and S. Dawkins, "Concepts and Terminology for Peer to Peer SIP", draft-ietf-p2psip-concepts-04 (work in progress), October 2011.

[I-D.ietf-p2psip-sip]
Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., Schulzrinne, H., and T. Schmidt, "A SIP Usage for RELOAD", draft-ietf-p2psip-sip-09 (work in progress), February 2013.

[RFC4353] Rosenberg, J., "A Framework for Conferencing with the Session Initiation Protocol (SIP)", RFC 4353, February 2006.

[landmarks-infocomm02]
Ratnasamy, ., Handley, ., Karp, ., and . Shenker, "Topologically-Aware Overlay Construction and Server Selection", Proc. of 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM ’02) pp. 1190-1199, 2002.

[timestamps-acsc88]
Fidge, C., "Timestamps in Message-Passing Systems that Preserve the Partial Ordering", Proceedings of 11th Australian Computer Science Conference, pp. 56-66, February 1988.

Appendix A. Change Log

The following changes have been made from version draft-knauf-p2psip-disco-04.

1. Editorial improvements.

2. Updated references.

The following changes have been made from version draft-knauf-p2psip-disco-03.

1. Adapted mechanisms for storing DisCo-Registrations to new requirements of Shared Resources draft [I-D.knauf-p2psip-share]

The following changes have been made from version draft-knauf-p2psip-disco-02.

1. DisCo-Registration uses now only the USER-CHAIN-ACL access control policy.

2. Adapted mechanisms for storing DisCo-Registrations to new requirements of Shared Resources draft [I-D.knauf-p2psip-share]

The following changes have been made from version draft-knauf-p2psip-disco-01.

1. The conference registration is now based on the Shared Resources draft [I-D.knauf-p2psip-share]:

   * DisCo-Registration Kind now meets the requirements for ShaRe.

   * Conference creation procedure now uses the ShaRe Access List.

   * Replaced USER-CHAIN-MATCH access policy for DisCo-Registration. Now uses USER-CHAIN-ACL or USER-PATTERN-MATCH.

2. Allow focus peers behind NAT

3. Added a ’node-id’ attribute to the event package XML <version> element.

4. Added a ’node-id’ attribute to the event package XML <focus> element.

5. Added a ’coordinate’ child element to the event package XML <focus> element.

6. Corrected typos/wording

The following changes have been made from version draft-knauf-p2psip-disco-00.

1. Updated references.

2. Corrected typos.

3. New Section: Conference State Synchronization

4. XML Event Package for Distributed Conferences

5. New mechanism for generating chained conference certificates

6. Allow shared writing of resources using Access Control Policy USER-CHAIN-MATCH

7. Media Negotiation mechanism in a distributed conference

8. New Section: Distributed Conference Control with SIP

Authors' Addresses

Alexander Knauf
HAW Hamburg
Berliner Tor 7
Hamburg D-20099
Germany

Phone: +4940428758067
Email: alexanderknauf@gmail.com

Thomas C. Schmidt
HAW Hamburg
Berliner Tor 7
Hamburg D-20099
Germany

Email: schmidt@informatik.haw-hamburg.de
URI: http://inet.cpt.haw-hamburg.de/members/schmidt

Gabriel Hege
daviko GmbH
Am Borsigturm 50
Berlin D-13507
Germany

Phone: +493043004344
Email: hege@daviko.com

Matthias Waehlisch
link-lab & FU Berlin
Hoenower Str. 35
Berlin D-10318
Germany

Email: mw@link-lab.net
URI: http://www.inf.fu-berlin.de/~waehl

P2PSIP
Internet-Draft
Intended status: Standards Track
Expires: January 16, 2014

N. Zong, Ed.
X. Jiang
R. Even
Huawei Technologies
Y. Zhang
CoolPad

July 15, 2013

An extension to RELOAD to support Direct Response Routing
draft-ietf-p2psip-drr-08

Abstract

This document proposes an optional extension to RELOAD to support direct response routing mode. RELOAD recommends symmetric recursive routing for routing messages. The new optional extension provides a shorter route for responses reducing the overhead on intermediate peers and describes the potential cases where this extension can be used.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

RELOAD [I-D.ietf-p2psip-base] recommends symmetric recursive routing (SRR) for routing messages and describes the extensions that would be required to support additional routing algorithms. Other than SRR, two other routing options: direct response routing (DRR) and relay peer routing (RPR) are also discussed in Appendix A of [I-D.ietf-p2psip-base]. As we show in section 3, DRR is advantageous over SRR in some scenarios by reducing load (CPU and link bandwidth) on intermediate peers. For example, in a closed network where every peer is in the same address realm, DRR performs better than SRR. In other scenarios, using a combination of DRR and SRR together is more likely to bring benefits than if SRR is used alone.

Note that in this document, we focus on DRR routing mode and its extensions to RELOAD to produce a standalone solution. Please refer to RPR draft [I-D.ietf-p2psip-rpr] for RPR routing mode.

We first discuss the problem statement in Section 3, then how to combine DRR and SRR is presented in Section 4. In Section 5, we give comparison on the cost of SRR and DRR in both managed and open networks. An extension to RELOAD to support DRR is proposed in Section 6. Some optional methods to check peer connectivity are introduced in Appendix A.

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 RFC 2119 [RFC2119].

We use the terminology and definitions from the RELOAD base draft [I-D.ietf-p2psip-base] extensively in this document. We also use terms defined in NAT behavior discovery [RFC5780]. Other terms used in this document are defined inline when used and are also defined below for reference.

Publicly Reachable: A peer is publicly reachable if it can receive unsolicited messages from any other peer in the same overlay. Note: "publicly" does not mean that the peers must be on the public Internet, because the RELOAD protocol may be used in a closed system.

Direct Response Routing (DRR): refers to a routing mode in which responses to P2PSIP requests are returned to the sending peer directly from the destination peer based on the sending peer’s own local transport address(es). For simplicity, the abbreviation DRR is used instead in the rest of the document.

Symmetric Recursive Routing (SRR): refers to a routing mode in which responses follow the reverse path of the request to get to the sending peer. For simplicity, the abbreviation SRR is used instead in the rest of the document.

3. Overview

RELOAD is expected to work under a great number of application scenarios. The situations where RELOAD is to be deployed differ greatly. For instance, some deployments are global, such as a Skype-like system intended to provide public service, while others run in closed networks of small scale. SRR works in any situation, but DRR may work better in some specific scenarios.

3.1. SRR and DRR

RELOAD is a simple request-response protocol. After sending a request, a peer waits for a response from a destination peer. There are several ways for the destination peer to send a response back to the source peer. In this section, we will provide detailed information on two routing modes: SRR and DRR.

Some assumptions are made in the following illustrations.

1) Peer A sends a request destined to a peer who is the responsible peer for Resource-ID k;

2) Peer X is the root peer being responsible for resource k;

3) The intermediate peers for the path from A to X are peer B, C, D.

3.1.1. Symmetric Recursive Routing (SRR)

For SRR, when the request sent by peer A is received by an intermediate peer B, C or D, each intermediate peer will insert information on the peer from whom they got the request in the via-list as described in RELOAD. As a result, the destination peer X will know the exact path which the request has traversed. Peer X will then send back the response in the reverse path by constructing a destination list based on the via-list in the request. Figure 1 illustrates SRR.

```
A            B            C             D           X
|  Request   |            |            |            |
|----------->|            |            |            |
|            | Request    |            |            |
|            |----------->|            |            |
|            |            | Request    |            |
```

```
   |            |            |----------->|            |
   |            |            |            | Request    |
   |            |            |            |----------->|
   |            |            |            |            |
   |            |            |            |  Response  |
   |            |            |            |<-----------|
   |            |            |  Response  |            |
   |            |            |<-----------|            |
   |            |  Response  |            |            |
   |            |<-----------|            |            |
   |  Response  |            |            |            |
   |<-----------|            |            |            |
   |            |            |            |            |
```

Figure 1. SRR routing mode

SRR works in any situation, especially when there are NATs or firewalls. A downside of this solution is that the message takes several hops to return to the peer, increasing the bandwidth usage and CPU/battery load of multiple peers.

3.1.2. Direct Response Routing (DRR)

In DRR, peer X receives the request sent by peer A through intermediate peer B, C and D, as in SRR. However, peer X sends the response back directly to peer A based on peer A's local transport address. In this case, the response is not routed through intermediate peers. Figure 2 illustrates DRR. Using a shorter route means less overhead on intermediate peers, especially in the case of wireless networks where the CPU and uplink bandwidth is limited. For example, in the absence of NATs, or if the NAT implements endpoint-independent filtering, this is the optimal routing technique. Note that establishing a secure connection requires multiple round trips. Please refer to Section 5 for cost comparison between SRR and DRR.

```
   A            B            C             D           X
   |  Request   |            |            |            |
   |----------->|            |            |            |
   |            | Request    |            |            |
   |            |----------->|            |            |
   |            |            | Request    |            |
   |            |            |----------->|            |
   |            |            |            | Request    |
   |            |            |            |----------->|
   |            |            |            |            |
   |            |            |            |  Response  |
   |<-----------+------------+------------+------------|
```

```
|            |            |            |            |
```

Figure 2. DRR routing mode

3.2. Scenarios where DRR can be used

This section lists several scenarios where using DRR would work, and identifies when the increased efficiency would be advantageous.

3.2.1. Managed or closed P2P systems

The properties that make P2P technology attractive, such as the lack of need for centralized servers, self-organization, etc. are attractive for managed systems as well as unmanaged systems. Many of these systems are deployed on private networks where peers are in the same address realm and/or can directly route to each other. In such a scenario, the network administrator can indicate preference for DRR in the peer’s configuration file. Peers in such a system would always try DRR first, but peers MUST also support SRR in case DRR fails. If during the process of establishing a direct connection with the sending peer, the responding peer receives a response with SRR as the preferred routing mode (or it fails to establish the direct connection), the responding peer SHOULD NOT use DRR but switch to SRR. The simple policy is to try DRR and if fails switch to SRR for all connections. A finer grained policy is when a peer keeps a list of unreachable peers based on trying DRR and use only SRR for these peers. The advantage in using DRR is on the network stability since it puts less overhead on the intermediate peers that will not route the responses. The intermediate peers will need to route less messages and save CPU resources as well as the link bandwidth usage.

3.2.2. Wireless scenarios

In some mobile deployments, using DRR may help with reducing radio battery usage and bandwidth by the intermediate peers. The service provider may recommend using DRR based on his knowledge of the topology.

4. Relationship between SRR and DRR

4.1. How DRR works

DRR is very simple. The only requirement is for the source peers to provide their potential (publically reachable) transport address to the destination peers, so that the destination peer knows where to send the response. Responses are sent directly to the requesting peer.

4.2. How SRR and DRR work together

DRR is not intended to replace SRR. It is better to use these two modes together to adapt to each peer's specific situation. In this section, we give some informative suggestions on how to transition between the routing modes in RELOAD.

According to base draft [I-D.ietf-p2psip-base], SRR MUST be supported. An overlay MAY be configured to use alternative routing algorithms, and alternative routing algorithms MAY be selected on a per-message basis. I.e., a node in an overlay which supports SRR and some other routing algorithm, for example DRR, might use SRR some of the time and DRR some of the time. A node joining the overlay should get from the configuration file the preferred routing mode. If an overlay runs within a private network and all peers in the system can reach each other directly, peers MAY send most of the transactions with DRR. On the contrary, using DRR SHOULD be discouraged in the open Internet or if the administrator does not feel he have enough information about the overlay network topology. A new overlay configuration element specifying the usage of DRR is defined in Section 7.

Alternatively, a peer can collect statistical data on the success of the different routing modes based on previous transactions and keep a list of non-reachable addresses. Based on this data, the peer will have a clearer view about the success rate of different routing modes. Other than the success rate, the peer can also get data of finer granularity, for example, the number of retransmission the peer needs to achieve a desirable success rate.

A typical strategy for the peer is as follows. A peer chooses to start with DRR based on the configuration. Based on the success rate seen from the lost message statistics or responses that used DRR, the peer can either continue to offer DRR first or switch to SRR. Note that a peer should use the DRR success statistic to decide if to continue using DRR or fall back to SRR. It is not recommended to make such decision per specific connection but this is an application decision.

5. Comparison on cost of SRR and DRR

The major advantages in using DRR are in going through less intermediate peers on the response. By doing that it reduces the load on those peers' resources like processing and communication bandwidth.

5.1. Closed or managed networks

As described in Section 3, many P2P systems run in a closed or managed environment (e.g. carrier networks) so that network administrators would know that they could safely use DRR.

SRR brings out more routing hops than DRR. Assuming that there are N peers in the P2P system and Chord is applied for routing, the number of hops for a response in SRR and DRR are listed in the following table. Establishing a secure connection between the sending peer and the responding peer with (D)TLS requires multiple messages. Note that establishing (D)TLS secure connections for P2P overlay is not optimal in some cases, e.g. direct response routing where (D)TLS is heavy for temporary connections. Instead, some alternate security techniques, e.g. using public keys of the destination to encrypt the messages, and signing timestamps to prevent reply attacks can be adopted. Therefore, in the following table, we show the cases of: 1) no (D)TLS in DRR; 2) still using DTLS in DRR as sub-optimal. As the worst-cost case, 7 messages are used during the DTLS handshaking [DTLS]. (TLS Handshake is a two round-trip negotiation protocol while DTLS handshake is a three round-trip negotiation protocol.)

Mode	Success	No. of Hops	No. of Msgs
SRR	Yes	log(N)	log(N)
DRR	Yes	1	1
DRR(DTLS)	Yes	1	7+1

Table 1. Comparison of SRR and DRR in closed networks

From the above comparison, it is clear that:

1) In most cases when N > 2 (2^1), DRR uses fewer hops than SRR. Using a shorter route means less overhead and resource usage on intermediate peers, which is an important consideration for adopting DRR in the cases where the resources such as CPU and bandwidth are limited, e.g. the case of mobile, wireless networks.

2) In the cases when N > 256 (2^8), DRR also uses fewer messages than SRR.

3) In the cases when N < 256, DRR uses more messages than SRR (but still uses fewer hops than SRR). So the consideration on whether using DRR or SRR depends on other factors like using less resources (bandwidth and processing) from the intermediate peers. Section 4 provides use cases where DRR has better chance to work or where the intermediary resources considerations are important.

5.2. Open networks

In open networks where DRR is not guaranteed to work, DRR can fall back to SRR if it fails after trial, as described in Section 4. Based on the same settings in Section 5.1, the number of hops, number of messages for a response in SRR and DRR are listed in the following table.

Mode	Success	No. of Hops	No. of Msgs
SRR	Yes	log(N)	log(N)
DRR	Yes	1	1
	Fail&Fall back to SRR	1+log(N)	1+log(N)
DRR(DTLS)	Yes	1	7+1
	Fail&Fall back to SRR	1+log(N)	8+log(N)

Table 2. Comparison of SRR and DRR in open networks

From the above comparison, it can be observed that trying to first use DRR could still provide an overall number of hops lower than directly using SRR. Suppose that P peers are publicly reachable, the number of hops in DRR and SRR is P*1+(N-P)*(1+logN), N*logN, respectively. The condition for fewer hops in DRR is P*1+(N-P)*(1+logN) < N*logN, which is P/N > 1/logN. This means that when the number of peers N grows, the required ratio of publicly reachable peers P/N for fewer hops in DRR decreases. Therefore, the chance of trying DRR with fewer hops than SRR becomes better as the scale of the network increases.

6. DRR extensions to RELOAD

Adding support for DRR requires extensions to the current RELOAD protocol. In this section, we define the extensions required to the protocol, including extensions to message structure and to message processing.

6.1. Basic requirements

All peers MUST be able to process requests for routing in SRR, and MAY support DRR routing requests.

6.2. Modification to RELOAD message structure

RELOAD provides an extensible framework to accommodate future extensions. In this section, we define a ForwardingOption structure to support DRR mode. Additionally we present a state-keeping flag to inform intermediate peers if they are allowed to not maintain state for a transaction.

6.2.1. State-keeping flag

RELOAD allows intermediate peers to maintain state in order to implement SRR, for example for implementing hop-by-hop retransmission. If DRR is used, the response will not follow the reverse path, and the state in the intermediate peers will not be cleared until such state expires. In order to address this issue, we propose a new flag, state-keeping flag, in the message header to indicate whether the state keeping is required in the intermediate peers.

flag : 0x08 IGNORE-STATE-KEEPING

If IGNORE-STATE-KEEPING is set, any peer receiving this message and which is not the destination of the message SHOULD forward the message with the full via_list and SHOULD NOT maintain any internal state.

6.2.2. Extensive routing mode

This draft introduces a new forwarding option for an extensive routing mode. This option conforms to the description in section 6.3.2.3 of [I-D.ietf-p2psip-base].

We first define a new type to define the new option, extensive_routing_mode:

The option value is illustrated in the following figure, defining the ExtensiveRoutingModeOption structure:

```
enum {(0),DRR(1),(255)} RouteMode;
struct {
        RouteMode               routemode;
        OverlayLinkType         transport;
        IpAddressPort           ipaddressport;
        Destination             destinations<1..2^8-1>;
} ExtensiveRoutingModeOption;
```

The above structure reuses OverlayLinkType, Destination and IpAddressPort structure defined in section 6.5.1.1, 6.3.2.2 and 6.3.1.1 of [I-D.ietf-p2psip-base].

RouteMode: refers to which type of routing mode is indicated to the destination peer.

OverlayLinkType: refers to the transport type which is used to deliver responses from the destination peer to the sending peer.

IpAddressPort: refers to the transport address that the destination peer use to send the response to. This will be a sending peer address for DRR.

Destination: refers to the sending peer itself. If the routing mode is DRR, then the destination only contains the sending peer's Node-ID.

6.3. Creating a request

6.3.1. Creating a request for DRR

When using DRR for a transaction, the sending peer MUST set the IGNORE-STATE-KEEPING flag in the ForwardingHeader. Additionally, the peer MUST construct and include a ForwardingOptions structure in the ForwardingHeader. When constructing the ForwardingOption structure, the fields MUST be set as follows:

1) The type MUST be set to extensive_routing_mode.

2) The ExtensiveRoutingModeOption structure MUST be used for the option field within the ForwardingOptions structure. The fields MUST be defined as follows:

2.1) routemode set to 0x01 (DRR).

2.2) transport set as appropriate for the sender.

2.3) ipaddressport set to the peer's associated transport address.

2.4) The destination structure MUST contain one value, defined as type node and set with the sending peer's own values.

6.4. Request and response processing

This section gives normative text for message processing after DRR is introduced. Here, we only describe the additional procedures for supporting DRR. Please refer to [I-D.ietf-p2psip-base] for RELOAD base procedures.

6.4.1. Destination peer: receiving a request and sending a response

When the destination peer receives a request, it will check the options in the forwarding header. If the destination peer can not understand extensive_routing_mode option in the request, it MUST attempt to use SRR to return an "Error_Unknown_Extension" response (defined in Section 6.3.3.1 and Section 14.9 of [I-D.ietf-p2psip-base]) to the sending peer.

If the routing mode is DRR, the peer MUST construct the Destination list for the response with only one entry, using the sending peer's Node-ID from the option in the request as the value.

In the event that the routing mode is set to DRR and there is not exactly one destination, the destination peer MUST try to return an "Error_Unknown_Extension" response (defined in Section 6.3.3.1 and Section 14.9 of [I-D.ietf-p2psip-base]) to the sending peer using SRR.

After the peer constructs the destination list for the response, it sends the response to the transport address which is indicated in the ipaddressport field in the option using the specific transport mode in the Forwardingoption. If the destination peer receives a retransmit with SRR preference on the message it is trying to respond to now, the responding peer SHOULD abort the DRR response and use SRR.

6.4.2. Sending peer: receiving a response

Upon receiving a response, the peer follows the rules in [I-D.ietf-p2psip-base]. The peer SHOULD note if DRR worked in order to decide if to offer DRR again. If the peer does not receive a response until the timeout it SHOULD resend the request using SRR.

7. Overlay configuration extension

This document extends the RELOAD overlay configuration by adding one new element, "route-mode", inside each "configuration" element.

The Compact Relax NG Grammar for this element is:

```
namespace route-mode = "urn:ietf:params:xml:ns:p2p:route-mode"

parameter &= element route-mode:mode { xsd:string }?
```

This namespace is added into the <mandatory-extension> element in the overlay configuration file. The defined routing modes include DRR and RPR.

Mode can be DRR or RPR and if specified in the configuration should be the preferred routing mode used by the application.

8. Security considerations

As a routing alternative, the security part of DRR conforms to section 13.6 of the base draft [I-D.ietf-p2psip-base] which describes routing security. The DRR routing option provide the information about the route back to the source. According to section 13 of the base drat the forwarding header MUST be digitally signed protecting the DRR routing information.

9. IANA considerations

9.1. A new RELOAD forwarding option

A new RELOAD Forwarding Option type is added to the Forwarding Option Registry defined in [I-D.ietf-p2psip-base].

Type: 0x02 - extensive_routing_mode

9.2. A new IETF XML registry

This section requests IANA to register the following URN in the "XML Namespaces" class of the "IETF XML Registry" in accordance with [RFC3688].

URI: urn:ietf:params:xml:ns:p2p:route-mode

Registrant Contact: The IESG

XML: This specification

10. Acknowledgements

David Bryan has helped extensively with this document, and helped provide some of the text, analysis, and ideas contained here. The authors would like to thank Ted Hardie, Narayanan Vidya, Dondeti Lakshminath, Bruce Lowekamp, Stephane Bryant, Marc Petit-Huguenin and Carlos Jesus Bernardos Cano for their constructive comments.

11. References

11.1. Normative references

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC2119, 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-26 (work in progress), February 2013.

[RFC3688] Mealling, M., "The IETF XML Registry ", BCP 81, RFC3688, January 2004.

11.2. Informative references

[DTLS] Modadugu, N., Rescorla, E., "The Design and Implementation of Datagram TLS", 11th Network and Distributed System Security Symposium (NDSS), 2004.

[RFC5780] MacDonald, D. and B. Lowekamp, "NAT Behavior Discovery Using STUN", RFC5780, May 2010.

[I-D.ietf-p2psip-rpr] Zong, N., Jiang, X., Even, R. and Zhang, Y., "An extension to RELOAD to support Relay Peer Routing", draft-ietf-p2psip-rpr-07, June 2013.

[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-04 (work in progress), October 2011.

[IGD2] UPnP Forum, "WANIPConnection:2 Service (http://upnp.org/specs/gw/UPnP-gw-WANIPConnection-v2-Service.pdf)", September 2010.

[RFC6886] Cheshire, S., Krochmal M., and K. Sekar, "NAT Port Mapping Protocol (NAT-PMP)", RFC6886, April 2013.

[RFC3424] Daigle, L., "IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation", RFC3424, November 2002.

12. References

Appendix A. Optional methods to investigate peer connectivity

This section is for informational purposes only for providing some mechanisms that can be used when the configuration information does not specify if DRR can be used. It summarizes some methods which can be used for a peer to determine its own network location compared

with NAT. These methods may help a peer to decide which routing mode it may wish to try. Note that there is no foolproof way to determine if a peer is publically reachable, other than via out-of-band mechanisms. For discussion about issues with address evaluation also see UNSAF [RFC3424]. As such, peers using these mechanisms may be able to optimize traffic, but must be able to fall back to SRR routing if the other routing mechanisms fail.

For DRR to function correctly, a peer may attempt to determine whether it is publicly reachable. If it is not, the peers should fall back to SRR. If the peer believes it is publically reachable, DRR may be attempted. NATs and firewalls are two major contributors preventing DRR from functioning properly. There are a number of techniques by which a peer can get its reflexive address on the public side of the NAT. After obtaining the reflexive address, a peer can perform further tests to learn whether the reflexive address is publicly reachable. If the address appears to be publicly reachable, the peers to which the address belongs can use DRR for responses.

Some conditions are unique in P2PSIP architecture which could be leveraged to facilitate the tests. In P2P overlay network, each peer only has partial a view of the whole network, and knows of a few peers in the overlay. P2P routing algorithms can easily deliver a request from a sending peer to a peer with whom the sending peer has no direct connection. This makes it easy for a peer to ask other peers to send unsolicited messages back to the requester.

In the following sections, we first introduce several ways for a peer to get the addresses needed for further tests. Then a test for learning whether a peer may be publicly reachable is proposed.

A.1. Getting addresses to be used as candidates for DRR

In order to test whether a peer may be publicly reachable, the peer should first get one or more addresses which will be used by other peers to send him messages directly. This address is either a local address of a peer or a translated address which is assigned by a NAT to the peer.

STUN is used to get a reflexive address on the public side of a NAT with the help of STUN servers. Discovery of NAT behavior using STUN is specified in [RFC5780]. Under RELOAD architecture, a few infrastructure servers can be leveraged for discovering NAT behavior, such as enrollment servers, diagnostic servers, bootstrap servers, etc.

The peer can use a STUN Binding request to one of STUN servers to trigger a STUN Binding response which returns the reflexive address from the server’s perspective. If the reflexive transport address is the same as the source address of the Binding request, the peer can determine that there likely is no NAT between it and the chosen infrastructure server (Certainly, in some rare cases, the allocated address happens to be the same as the source address. Further tests will detect this case and rule it out in the end.). Usually, these infrastructure severs are publicly reachable in the overlay, so the peer can be considered publicly reachable. On the other hand, with the techniques in [RFC5780], a peer can also decide whether it is behind a NAT with endpoint-independent mapping behavior. If the peer is behind a NAT with endpoint- independent mapping behavior, the reflexive address should also be a candidate for further tests.

UPnP-IGD [IGD2] is a mechanism that a peer can use to get the assigned address from its residential gateway and after obtaining this address to communicate it with other peers, the peer can receive unsolicited messages from outside, even though it is behind a NAT. So the address obtained through the UPnP mechanism should also be used for further tests.

Another way that a peer behind NAT can use to learn its assigned address by NAT is NAT-PMP [RFC6886]. Like in UPnP-IGD, the address obtained using this mechanism should also be tested further.

The above techniques are not exhaustive. These techniques can be used to get candidate transport addresses for further tests.

A.2. Public reachability test

Using the transport addresses obtained by the above techniques, a peer can start a test to learn whether the candidate transport address is publicly reachable. The basic idea for the test is for a peer to send a request and expect another peer in the overlay to send back a response. If the response is received by the sending peer successfully and also the peer giving the response has no direct connection with the sending peer, the sending peer can determine that the address is probably publicly reachable and hence the peer may be publicly reachable at the tested transport address.

In a P2P overlay, a request is routed through the overlay and finally a destination peer will terminate the request and give the response. In a large system, there is a high probability that the destination peer has no direct connection with the sending peer. Especially in RELOAD architecture, every peer maintains a connection table. So it is easier for a peer to check whether it has direct connection with another peer.

If a peer wants to test whether its transport address is publicly reachable, it can send a request to the overlay. The routing for the test message would be different from other kinds of requests because it is not for storing/fetching something to/from the overlay or locating a specific peer, instead it is to get a peer who can deliver the sending peer an unsolicited response and which has no direct connection with him. Each intermediate peer receiving the request first checks whether it has a direct connections with the sending peer. If there is a direct connection, the request is routed to the next peer. If there is no direct connection, the intermediate peer terminates the request and sends the response back directly to the sending peer with the transport address under test.

After performing the test, if the peer determines that it may be publicly reachable, it can try DRR in subsequent transactions.

Authors' Addresses

Ning Zong (editor)
Huawei Technologies

Email: zongning@huawei.com

Xingfeng Jiang
Huawei Technologies

Email: jiang.x.f@huawei.com

Roni Even
Huawei Technologies

Email: roni.even@mail01.huawei.com

Yunfei Zhang
CoolPad

Email: hishigh@gmail.com

P2PSIP
Internet-Draft
Intended status: Standards Track
Expires: January 16, 2014

C. Jennings
Cisco
B. Lowekamp
Skype
E. Rescorla
RTFM, Inc.
S. Baset
H. Schulzrinne
Columbia University
T C. Schmidt, Ed.
HAW Hamburg
July 15, 2013

A SIP Usage for RELOAD
draft-ietf-p2psip-sip-10

Abstract

This document defines a SIP Usage for REsource LOcation And Discovery (RELOAD). The SIP Usage provides the functionality of a SIP proxy or registrar in a fully-distributed system and includes a lookup service for Address of Records (AORs) stored in the overlay. It also defines Globally Routable User Agent Uris (GRUUs) that allow the registrations to map an AOR to a specific node reachable through the overlay. After such initial contact of a peer, the AppAttach method is used to establish a direct connection between nodes through which SIP messages are exchanged.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

The REsource LOcation And Discovery (RELOAD) [I-D.ietf-p2psip-base] specifies a peer-to-peer (P2P) signaling protocol for the general use on the Internet. This document defines a SIP Usage of RELOAD that allows SIP [RFC3261] user agents (UAs) to establish peer-to-peer SIP (or SIPS) sessions without the requirement for permanent proxy or registration servers, e.g., a fully distributed telephony service. In such a network, the RELOAD overlay itself performs the registration and rendezvous functions ordinarily associated with such servers.

The SIP Usage involves two basic functions.

Registration: SIP UAs can use the RELOAD data storage functionality to store a mapping from their address-of-record (AOR) to their Node-ID in the overlay, and to retrieve the Node-ID of other UAs.

Rendezvous: Once a SIP UA has identified the Node-ID for an AOR it wishes to call, it can use the RELOAD message routing system to set up a direct connection for exchanging SIP messages.

Mappings are stored in the SipRegistration Resource Record defined in this document. All operations required to perform a SIP registration or rendezvous are standard RELOAD protocol methods.

For example, Bob registers his AOR, "bob@dht.example.com", for his Node-ID "1234". When Alice wants to call Bob, she queries the overlay for "bob@dht.example.com" and receives Node-ID 1234 in return. She then uses the overlay routing to establish a direct connection with Bob and can directly transmit a standard SIP INVITE. In detail, this works along the following steps.

1. Bob, operating Node-ID 1234, stores a mapping from his AOR to his Node-ID in the overlay by applying a Store request for "bob@dht.example.com -> 1234".
2. Alice, operating Node-ID 5678, decides to call Bob. She retrieves Node-ID "1234" by performing a Fetch request on "bob@dht.example.com".
3. Alice uses the overlay to route an AppAttach message to Bob’s peer (ID 1234). Bob responds with his own AppAttach and they set up a direct connection, as shown in Figure 1. Note that mutual ICE checks are invoked automatically from AppAttach message exchange.

```
                          Overlay
            Alice       Peer1     ...          PeerN      Bob
            (5678)                                      (1234)
            ------------------------------------------------
            AppAttach ->
                        AppAttach ->
                                  AppAttach ->
                                              AppAttach ->
                                                 <- AppAttach
                                       <- AppAttach
                             <- AppAttach
                 <- AppAttach

            <------------------ ICE Checks ----------------->
            INVITE ------------------------------------------>
            <------------------------------------------- OK
            ACK --------------------------------------------->
            <------------ ICE Checks for media ------------->
            <-------------------- RTP ----------------------->
```

Figure 1: Connection setup in P2P SIP using the RELOAD overlay

It is important to note that here the only role of RELOAD is to set up the direct SIP connection between Alice and Bob. As soon as the ICE checks complete and the connection is established, ordinary SIP or SIPS is used. In particular, the establishment of the media channel for a phone call happens via the usual SIP mechanisms, and RELOAD is not involved. Media never traverses the overlay. After the successful exchange of SIP messages, call peers run ICE connectivity checks for media.

In addition to mappings from AORs to Node-IDs, the SIP Usage also allows mappings from AORs to other AORs. This enables an indirection useful for call forwarding. For instance, if Bob wants his phone calls temporarily forwarded to Charlie, he can store the mapping "bob@dht.example.com -> charlie@dht.example.com". When Alice wants to call Bob, she retrieves this mapping and can then fetch Charlie’s AOR to retrieve his Node-ID. These mechanisms are described in Section 3.

Alternatively, Globally Routable User Agent URIs (GRUUs) can be used for directly accessing peers. They are handled via a separate mechanism, as described in Section 6.

The SIP Usage for RELOAD addresses a fully distributed deployment of session-based services among overlay peers. Two opposite scenarios of deploying P2P SIP services are in the focus of this document: A

highly regulated environment of a "single provider" that admits parties using AORs with domains from controlled namespace(s), only, and an open, multi-party infrastructure that liberally allows a registration and rendezvous for various or any domain namespace. It is noteworthy in this context that - in contrast to regular SIP - domain names play no role in routing to a proxy server. Once connectivity to an overlay is given, any name registration can be technically processed.

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 RFC 2119 [RFC2119].

We use the terminology and definitions from Concepts and Terminology for Peer to Peer SIP [I-D.ietf-p2psip-concepts] and the RELOAD Base Protocol [I-D.ietf-p2psip-base] extensively in this document.

In addition, term definitions from SIP [RFC3261] apply to this memo. The term AOR is the SIP "Address of Record" used to identify a user in SIP. For example, alice@example.com could be the AOR for Alice. For the purposes of this specification, an AOR is considered not to include the scheme (e.g sip:) as the AOR needs to match the rfc822Name in the X509v3 certificates. It is worth noting that SIP and SIPS are distinguished in P2PSIP by the Application-ID.

3. Registering AORs in the Overlay

3.1. Overview

In ordinary SIP, a UA registers its AOR and location with a registrar. In RELOAD, this registrar function is provided by the overlay as a whole. To register its location, a RELOAD peer stores a SipRegistration Resource Record under its own AOR using the SIP-REGISTRATION Kind, which is formally defined in Section 7. A RELOAD overlay MAY restrict the storage of AORs. Namespaces (i.e., the right hand side of the AOR) that are supported for registration and lookup can be configured for each RELOAD deployment as described in Section 3.4.

As a simple example, consider Alice with AOR "alice@dht.example.org" at Node-ID "1234". She might store the mapping "alice@dht.example.org -> 1234" telling anyone who wants to call her to contact node "1234".

RELOAD peers MAY store two kinds of SIP mappings,

- o from an AOR to a destination list (a single Node-ID is just a trivial destination list), or
- o from an AOR to another AOR.

The meaning of the first kind of mapping is "in order to contact me, form a connection with this peer." The meaning of the second kind of mapping is "in order to contact me, dereference this AOR". The latter allows for forwarding. For instance, if Alice wants her calls to be forwarded to her secretary, Sam, she might insert the following mapping "alice@dht.example.org -> sam@dht.example.org".

3.2. Data Structure

This section defines the SipRegistration Resource Record as follows:

```
enum { sip_registration_uri(1), sip_registration_route(2),
   (255) } SipRegistrationType;

select (SipRegistration.type) {
  case sip_registration_uri:
    opaque               uri<0..2^16-1>;

  case sip_registration_route:
    opaque               contact_prefs<0..2^16-1>;
    Destination          destination_list<0..2^16-1>;

  /* This type can be extended */

} SipRegistrationData;

struct {
   SipRegistrationType   type;
   uint16                length;
   SipRegistrationData   data;
} SipRegistration;
```

The contents of the SipRegistration Resource Record are:

type
the type of the registration

length
the length of the rest of the PDU

data
the registration data

- o If the registration is of type "sip_registration_uri", then the contents are an opaque string containing the URI.
- o If the registration is of type "sip_registration_route", then the contents are an opaque string containing the callee's contact preferences and a destination list for the peer.

The encoding of contact_prefs - the callee's contact preferences - follows the media feature set syntax of [RFC2533] (see also [RFC2738]). As an example, a voicemail server that is a UA that supports audio and video media types and is not mobile would carry the following feature set description in its contact_prefs attribute:

```
(& (sip.audio=TRUE)
      (sip.video=TRUE)
      (sip.actor=msg-taker)
      (sip.automata=TRUE)
      (sip.mobility=fixed)
      (| (sip.methods=INVITE) (sip.methods=BYE) (sip.methods=OPTIONS)
         (sip.methods=ACK) (sip.methods=CANCEL)))
```

A callee MAY indicate that it prefers contact via a particular SIP scheme - SIP or SIPS - by using one of the following contact_prefs attribute:

```
      (sip.schemes=SIP)
      (sip.schemes=SIPS)
```

RELOAD explicitly supports multiple registrations for a single AOR. The registrations are stored in a Dictionary with Node-IDs as the dictionary keys. Consider, for instance, the case where Alice has two peers:

- o her desk phone (1234)
- o her cell phone (5678)

Alice might store the following in the overlay at resource "alice@dht.example.com".

- o A SipRegistration of type "sip_registration_route" with dictionary key "1234" and value "1234".
- o A SipRegistration of type "sip_registration_route" with dictionary key "5678" and value "5678".

Note that this structure explicitly allows one Node-ID to forward to another Node-ID. For instance, Alice could set calls to her desk phone to ring at her cell phone by storing a SipRegistration of type "sip_registration_route" with dictionary key "1234" and value "5678".

3.3. Access Control

In order to prevent hijacking or other misuse, registrations are subject to access control rules. Two kinds of restrictions apply:

- o A Store is permitted only for AORs with domain names that fall into the namespaces supported by the RELOAD overlay instance.
- o Storing requests are performed according to the USER-NODE-MATCH access control policy of RELOAD.

Before issuing a Store request to the overlay, any peer SHOULD verify that the AOR of the request is a valid Resource Name with respect to its domain name and the namespaces defined in the overlay configuration document (see Section 3.4).

Before a Store is permitted, the storing peer MUST check that:

- o The AOR of the request is a valid Resource Name with respect to the namespaces defined in the overlay configuration document.
- o The certificate contains a username that is a SIP AOR which hashes to the Resource-ID it is being stored at.
- o The certificate contains a Node-ID that is the same as the dictionary key it is being stored at.

Note that these rules permit Alice to forward calls to Bob without his permission. However, they do not permit Alice to forward Bob’s calls to her. See Section 8.2.2 for additional descriptions.

3.4. Overlay Configuration Document Extension

The use of a SIP-enabled overlay MAY be restricted to users with AORs from specific domains. When deploying an overlay service, providers can decide about these use case scenarios by defining a set of namespaces for admissible domain names. This section extends the overlay configuration document by defining new elements for patterns that describe a corresponding domain name syntax.

A RELOAD overlay can be configured to accept store requests for any

AOR, or to apply domain name restrictions. For the latter, an enumeration of admissible domain names including wildcarded name patterns of the following form MAY be configured.

Example of Domain Patterns:
dht\.example\.com
.*\.my\.name

In this example, any AOR will be accepted that is either of the form <user>@dht.example.com, or ends with the domain "my.name". When restrictions apply and in the absence of domain patterns, the default behavior is to accept only AORs that exactly match the domain name of the overlay. Otherwise, i.e., when restrictions are not configured (attribute enable not set), the default behavior is to accept any AOR. In the absence of a <domain-restrictions> element, implementors SHOULD assume this default value. Encoding of the domain name complies to the restricted ASCII character set without character escaping as defined in Section 19.1 of [RFC3261].

The <domain-restrictions> element serves as a container for zero to multiple <pattern> sub-elements. A <pattern> element MAY be present if the "enable" attribute of its parent element is set to true. Each <pattern> element defines a pattern for constructing admissible resource names. It is of type xsd:string and interpreted as a regular expression according to "POSIX Extended Regular Expression" (see the specifications in [IEEE-Posix]).

The Relax NG Grammar for the AOR Domain Restriction reads:

```
<!-- AOR DOMAIN RESTRICTION URN SUB-NAMESPACE -->

namespace sip = "urn:ietf:params:xml:ns:p2p:config-base:sip"

<!--  AOR DOMAIN RESTRICTION ELEMENT -->

Kind-parameter &= element sip:domain-restriction {

    attribute enable { xsd:boolean }

    <!-- PATTERN ELEMENT -->

    element pattern { xsd:string }*
}?
```

4. Looking up an AOR

4.1. Finding a Route to an AOR

A RELOAD user, member of an overlay, who wishes to call another user with given AOR SHALL proceed in the following way.

AOR is GRUU? If the AOR is a GRUU for this overlay, the callee can be contacted directly as described in Section 6.

AOR domain is hosted in overlay? If the domain part of the AOR matches a domain pattern configured in the overlay, the user can continue to resolve the AOR in this overlay. The user MAY choose to query the DNS service records to search for additional support of this domain name.

AOR domain not supported by overlay? If the domain part of the AOR is not supported in the current overlay, the user SHOULD query the DNS (or other discovery services at hand) to search for an alternative overlay that services the AOR under request. Alternatively, standard SIP procedures for contacting the callee SHOULD be used.

AOR inaccessible? If all of the above contact attempts fail, the call fails.

The procedures described above likewise apply when nodes are simultaneously connected to several overlays.

4.2. Resolving an AOR

A RELOAD user that has discovered a route to an AOR in the current overlay SHALL execute the following steps.

1. Perform a Fetch for Kind SIP-REGISTRATION at the Resource-ID corresponding to the AOR. This Fetch SHOULD NOT indicate any dictionary keys, so that it will fetch all the stored values.
2. If any of the results of the Fetch are non-GRUU AORs, then repeat step 1 for that AOR.
3. Once only GRUUs and destination lists remain, the peer removes duplicate destination lists and GRUUs from the list and initiates SIP or SIPS connections to the appropriate peers as described in the following sections. If there are also external AORs, the peer follows the appropriate procedure for contacting them as well.

5. Forming a Direct Connection

5.1. Setting Up a Connection

Once the peer has translated the AOR into a set of destination lists, it then uses the overlay to route AppAttach messages to each of those peers. The "application" field MUST be either 5060 to indicate SIP or 5061 for using SIPS. If certificate-based authentication is in use, the responding peer MUST present a certificate with a Node-ID matching the terminal entry in the route list. Note that it is possible that the peers already have a RELOAD connection mutually established. This MUST NOT be used for SIP messages unless it is a SIP connection. A previously established SIP connection MAY be used for a new call.

Once the AppAttach succeeds, the peer sends plain or (D)TLS encrypted SIP messages over the connection as in normal SIP. A caller MAY choose to contact the callee using SIP or secure SIPS, but SHOULD follow a preference indicated by the callee in its contact_prefs attribute (see Section 3.2). A callee MAY choose to listen on both SIP and SIPS ports and accept calls from either SIP scheme, or select a single one. However, a callee that decides to accept SIPS calls, only, SHOULD indicate its choice by setting the corresponding attribute in its contact_prefs.

5.2. Keeping a Connection Alive

In many cases, RELOAD connections will traverse NATs and Firewalls that maintain states established from ICE [RFC5245] negotiations. It is the responsiblity of the application to provide sufficiently frequent traffic to keep NAT and Firewall states present and the connection alive. Thus an application SHOULD survey traffic pauses on each of its SIP or SIPs connections and connection-wise issue a RELOAD ping after each pause excceeding the STUN indication message interval.

6. Using GRUUs

Globally Routable User Agent Uris (GRUUs) [RFC5627] have been designed to allow direct routing without the indirection of a SIP proxy function. The concept is transferred to RELOAD overlays as follows. GRUUs in RELOAD are constructed by embedding a base64-encoded destination list in the gr URI parameter of the GRUU. The base64 encoding is done with the alphabet specified in table 1 of [RFC4648] with the exception that ˜ is used in place of =.
Example of a RELOAD GRUU:
alice@example.com;gr=MDEyMzQ1Njc4OTAxMjM0NTY3ODk˜

GRUUs do not require to store data in the Overlay Instance. Rather

when a peer needs to route a message to a GRUU in the same P2P overlay, it simply uses the destination list and connects to that peer. Because a GRUU contains a destination list, it MAY have the same contents as a destination list stored elsewhere in the resource dictionary.

Anonymous GRUUs [RFC5767] are constructed analogously, but require either that the enrollment server issues a different Node-ID for each anonymous GRUU required, or that a destination list be used that includes a peer that compresses the destination list to stop the Node-ID from being revealed.

7. SIP-REGISTRATION Kind Definition

This section defines the SIP-REGISTRATION Kind.

Name SIP-REGISTRATION

Kind IDs The Resource Name for the SIP-REGISTRATION Kind-ID is the AOR of the user. The data stored is a SipRegistration, which can contain either another URI or a destination list to the peer which is acting for the user.

Data Model The data model for the SIP-REGISTRATION Kind-ID is dictionary. The dictionary key is the Node-ID of the storing peer. This allows each peer (presumably corresponding to a single device) to store a single route mapping.

Access Control USER-NODE-MATCH. Note that this matches the SIP AOR against the rfc822Name in the X509v3 certificate. The rfc822Name does not include the scheme so that the "sip:" prefix needs to be removed from the SIP AOR before matching.

Data stored under the SIP-REGISTRATION Kind is of type SipRegistration. This comes in two varieties:

sip_registration_uri
: a URI which the user can be reached at.

sip_registration_route
: a destination list which can be used to reach the user’s peer.

8. Security Considerations

8.1. RELOAD-Specific Issues

This Usage for RELOAD does not define new protocol elements or operations. Hence no new threats arrive from message exchanges in RELOAD.

This document introduces an AOR domain restriction function that must be surveyed by the storing peer. A misconfigured or malicious peer could cause frequent rejects of illegitimate storing requests. However, domain name control relies on a lightweight pattern matching and can be processed prior to validating certificates. Hence no extra burden is introduced for RELOAD peers beyond loads already present in the base protocol.

8.2. SIP-Specific Issues

8.2.1. Fork Explosion

Because SIP includes a forking capability (the ability to retarget to multiple recipients), fork bombs are a potential DoS concern. However, in the SIP usage of RELOAD, fork bombs are a much lower concern than in a conventional SIP Proxy infrastructure, because the calling party is involved in each retargeting event. It can therefore directly measure the number of forks and throttle at some reasonable number.

8.2.2. Malicious Retargeting

Another potential DoS attack is for the owner of an attractive AOR to retarget all calls to some victim. This attack is common to SIP and difficult to ameliorate without requiring the target of a SIP registration to authorize all stores. The overhead of that requirement would be excessive and in addition there are good use cases for retargeting to a peer without its explicit cooperation.

8.2.3. Misuse of AORs

A RELOAD overlay and enrollment service that liberally accept registrations for AORs of domain names unrelated to the overlay instance and without further justification, eventually store presence state for misused AORs. An attacker could hijack names, register a bogus presence and attract calls dedicated to a victim that resides within or outside the Overlay Instance.

A hijacking of AORs can be mitigated by restricting the name spaces admissible in the Overlay Instance, or by additional verification actions of the enrollment service. To prevent an (exclusive) routing to a bogus registration, a caller can in addition query the DNS (or

other discovery services at hand) to search for an alternative presence of the callee in another overlay or a normal SIP infrastructure.

8.2.4. Privacy Issues

All RELOAD SIP registration data is public. Methods of providing location and identity privacy are still being studied. Location privacy can be gained from using anonymous GRUUs.

9. IANA Considerations

9.1. Data Kind-ID

IANA shall register the following code point in the "RELOAD Data Kind-ID" Registry (cf., [I-D.ietf-p2psip-base]) to represent the SIP-REGISTRATION Kind, as described in Section 7. [NOTE TO IANA/ RFC-EDITOR: Please replace RFC-AAAA with the RFC number for this specification in the following list.]

Kind	Kind-ID	RFC
SIP-REGISTRATION	1	RFC-AAAA

9.2. XML Name Space Registration

This document registers the following URI for the config XML namespace in the IETF XML registry defined in [RFC3688]

URI: urn:ietf:params:xml:ns:p2p:config-base:sip
Registrant Contact: The IESG
XML: N/A, the requested URI is an XML namespace

10. Acknowledgments

This document was generated in parts from initial drafts and discussions in the early specification phase of the P2PSIP base protocol. Significant contributions (in alphabetical order) were from David A. Bryan, James Deverick, Marcin Matuszewski, Jonathan Rosenberg, and Marcia Zangrilli, which is gratefully acknowledged.

Additional thanks go to all those who helped with ideas, discussions, and reviews, in particular (in alphabetical order) Michael Chen, Marc Petit-Huguenin, Brian Rosen, and Matthias Waehlisch.

11. References

11.1. 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-26 (work in progress), February 2013.

[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.

[RFC2533] Klyne, G., "A Syntax for Describing Media Feature Sets", RFC 2533, March 1999.

[RFC2738] Klyne, G., "Corrections to "A Syntax for Describing Media Feature Sets"", RFC 2738, December 1999.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5627] Rosenberg, J., "Obtaining and Using Globally Routable User Agent URIs (GRUUs) in the Session Initiation Protocol (SIP)", RFC 5627, October 2009.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[IEEE-Posix] "IEEE Standard for Information Technology - Portable Operating System Interface (POSIX) - Part 2: Shell and Utilities (Vol. 1)", IEEE Std 1003.2-1992, ISBN 1-55937-255-9, January 1993.

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-05 (work in progress), July 2013.

[RFC5767] Munakata, M., Schubert, S., and T. Ohba, "User-Agent-Driven Privacy Mechanism for SIP", RFC 5767, April 2010.

[I-D.ietf-p2psip-share]
Knauf, A., Schmidt, T., Hege, G., and M. Waehlisch, "A Usage for Shared Resources in RELOAD (ShaRe)", draft-ietf-p2psip-share-01 (work in progress), February 2013.

Appendix A. Third Party Registration

In traditional SIP, the mechanism of a third party registration (i.e., an assistant acting for a boss, changing users register a role-based AOR, ...) is defined in Section 10.2 of [RFC3261]. This is a REGISTER which uses the URI of the third-party in its From header and cannot be translated directly into a P2PSIP registration, because only the owner of the certificate can store a SIP-REGISTRATION in a RELOAD overlay.

A way to implement third party registration is by using the extended access control mechanism USER-CHAIN-ACL defined in [I-D.ietf-p2psip-share]. Creating a new Kind "SIP-3P-REGISTRATION" that is ruled by USER-CHAIN-ACL allows the owner of the certificate to delegate the right for registration to individual third parties. In this way, original SIP functionality can be regained without weakening the security control of RELOAD.

Appendix B. Change Log

B.1. Changes since draft-ietf-p2psip-sip-09

- o Added subsection on keepalive
- o Updated references

B.2. Changes since draft-ietf-p2psip-sip-08

- o Added the handling of SIPS
- o Specified use of Posix regular expressions in configuration document
- o Added IANA registration for namespace
- o Editorial polishing
- o Updated and extended references

B.3. Changes since draft-ietf-p2psip-sip-07

- o Cleared open issues
- o Clarified use cases after WG discussion
- o Added configuration document extensions for configurable domain names
- o Specified format of contact_prefs
- o Clarified routing to AORs
- o Extended security section
- o Added Appendix on Third Party Registration
- o Added IANA code points
- o Editorial polishing
- o Updated and extended references

B.4. Changes since draft-ietf-p2psip-sip-06

- o Added Open Issue

Authors' Addresses

Cullen Jennings
Cisco
170 West Tasman Drive
MS: SJC-21/2
San Jose, CA 95134
USA

Phone: +1 408 421-9990
Email: fluffy@cisco.com

Bruce B. Lowekamp
Skype
Palo Alto, CA
USA

Email: bbl@lowekamp.net

Eric Rescorla
RTFM, Inc.
2064 Edgewood Drive
Palo Alto, CA 94303
USA

Phone: +1 650 678 2350
Email: ekr@rtfm.com

Salman A. Baset
Columbia University
1214 Amsterdam Avenue
New York, NY
USA

Email: salman@cs.columbia.edu

Henning Schulzrinne
Columbia University
1214 Amsterdam Avenue
New York, NY
USA

Email: hgs@cs.columbia.edu

Thomas C. Schmidt (editor)
HAW Hamburg
Berliner Tor 7
Hamburg 20099
Germany

Email: schmidt@informatik.haw-hamburg.de