Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Anwendungen 2 - SoSe 2010

Alexander Knauf

RELOAD – Usages for P2P Data Storage and Discovery

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Contents

1 Introduction

IP telephony, also referred as Internet telephony or Voice over IP (VoIP), describes mechanisms that use the Internet Protocol (IP) to establish calls between two or more end-systems. In contrary to traditional land line telephony where each end-system is assigned to a fixed telephone number, IP addresses are changing every time a new connection is established. This fact makes IP addresses unusable to assign them as common identifier for VoIP services. Instead of this, a VoIP-enabled end-system registers its address at a dedicated server that is in charge of call establishment. Since the migration from traditional telephony to VoIP services is a slow-going process and the fact that client/server architectures do not scale to a very large size, initiated a peer-to-peer based VoIP solution. Because the most common signaling protocol for establishing VoIP call and multimedia conferences is the Session Initiation Protocol (SIP)[1], the area of peer-to-peer based communication is called P2PSIP.

The REsource LOcation And Discovery (RELOAD) [2] Internet draft is an almost completed IETF standard defining a P2PSIP for use on the Internet. It provides its clients an abstract data storage, messaging and connection service where all clients are in a cooperation of peers forming an overlay network. The service layer is thereby independent from the overlay algorithm by providing an abstract interface to the overlay layer. RELOAD is designed to support the session handling with the Session Initiation Protocol by providing mechanisms to store and resolve mappings for SIP URIs and establishment of secure transport connections while traversing Network Address Translations (NATs). RELOAD is thereby not restricted to a SIP usage, rather, it can be utilized other applications that have similar demands. Applications that want to deploy services on RELOAD must define a so called *Usage*. Usages define a set of specifications like data structures (called *Kinds*) or access policies that have to be implemented.

The features provided by RELOAD base protocol contributed the decision to phrase the *Infrastructure Independent Conferencing* [3] approach as new Usage for the RELOAD. Therefore, this document presents concurrency work to *Infrastructure Independent Conferencing* by firstly introducing the main functionalities and mechanisms of the RELOAD base protocol. Secondly, by presenting three applications that defined Usages that utilize RELOAD as platform.

The reminder of this document is structured as follows. Section 2 gives a short review on main topics of *Anwendungen 1* [3], followed by an introduction and architecture of the RELOAD protocol in section 3. Section 4 then presents three applications that defined Usages for RELOAD protocol. This document concludes and gives a view on future work in section 5.

2 P2P Conferencing Using Overlay Networks

Countering infrastructural dependences and single point of failure problems in traditional conferencing solutions, an approach for distributed conferencing that uses P2P overlay networks was proposed in the previous work *Anwendungen 1* [3]. The main concept is to 'virtualize' the conference URI within a P2P overlay network and to equally distribute conference control and media mixing among the conference participants. The Goal of virtualizing an URI is to separate a logical identifier from any physical instance. P2P overlays facilitate such a mapping by allowing its peers to store data values using an abstract data storage and lookup interface. As shown in figure 1, a conference is controlled by two peers, which are connected to a P2P overlay. Participants can be connected to one of these controllers that are called focus in SIP. Focus peers are have to negotiate media parameters,provide conference state notification services and are responsible to connect their clients to the media of the conference. A new participant can retrieve contact informations to one of the focus peer by querying the overlay to resolve the conference URI to retrieve IP addresses and location informations. The new participant then selects focus peers which is the relatively closed to itself in the network and initiates a SIP session.

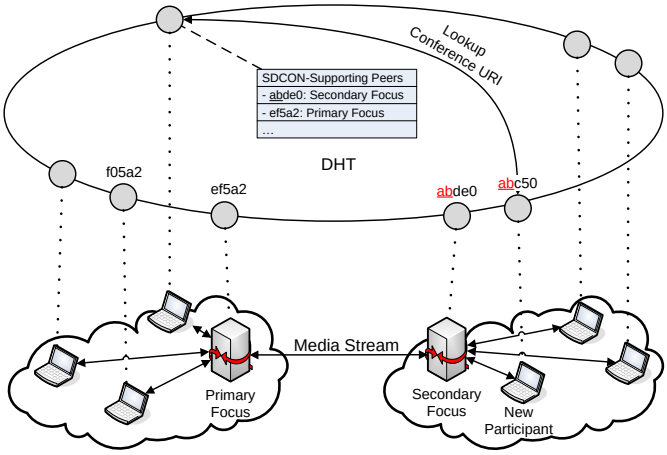

Figure 1: Discovery of a secondary focus using proximity information

The RELOAD base protocol complies to many demands for an infrastructure independent conferencing approach. This caused the decision to define a new RELOAD Usage for distributed conference control [4] and to present RELOAD and other Usages as concurrency work in this document.

3 Introducing RELOAD base protocol

RELOAD serves as a P2P service platform by providing a message transport protocol, data storage and lookup functionalities, as well as connection establishment for different types of applications. Figure 2 gives an overview of the main functionalities of the RELOAD base protocol. Since connectivity of many peers in an overlay can be limited by NATs or firewalls, Interactive Connectivity Establishment (ICE) [5] is supported for NAT and firewall traversal. RELOAD also provides a security framework based on public/private-key certificates to establish trust relations and message authentication. Overlay messages are designed with a simple and lightweight forwarding header reducing forwarding effort and increasing the routing performance. A noteworthy feature of RELOAD is that the overlay algorithm to be used is not fixed, but left to the implementation. However, the current version of the RELOAD draft foresees a deployment on an improved Chord [6] distributed hash table (DHT). To support different applications, RELOAD allows for the specification of new Usages. A Usage defines the data structures *(Kinds)* to be stored, the corresponding data identifier *(kind-ID)*, access control rules to those resources and how the resource overlay IDs are formed.

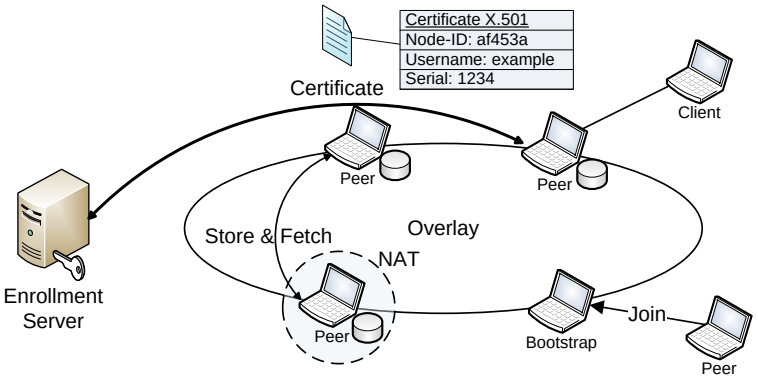

Figure 2: RELOAD — overview of functionalities

3.1 Architecture

The default structure of Internet protocols is given by a layered model [7] ordered from application layer down to link or physical layer. RELOAD defines a new Internet standard thus its conceptual architecture is given by the layered model in figure 3. Since RELOAD is settled at application layer the most bottom link layer actually is the transport layer in the real Internet model.

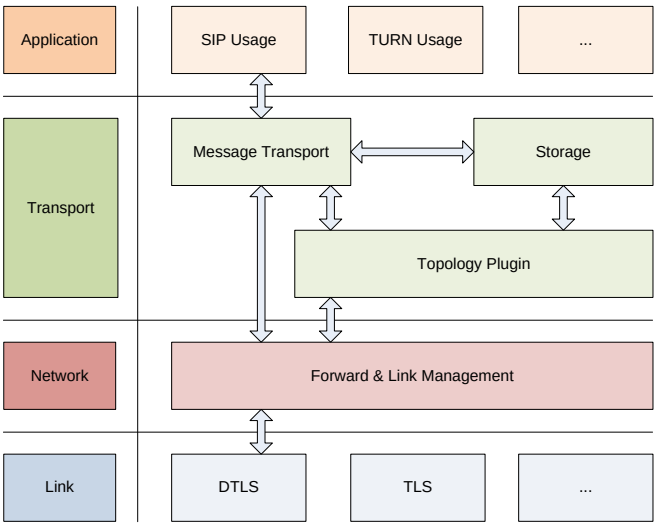

Figure 3: RELOAD – architecture

Application Layer The top layer in RELOAD is called Usage Layer. Every application that uses RELOAD as abstract message transport and storage service must define a set of application-specific requirements within a Usage. It represents a template for developers for implementing a RELOAD network interface. Applications are not restricted to only define or use one Usage, instead scenarios are allowed where are Usages utilizes another Usage. For instance, a SIP Usage could use a Usage for voice-mail as additional service. Each Usage is associated with an access control policy that defines whether a storing request from a given node will succeed or fail.

Transport Layer The RELOAD specification actually begins at the transport layer by defining components for *Message Transport*, *Storage* and the *Topology Plugin*.

The message transport component provides a generic interface for message routing and is responsible for the end-to-end communication between participating RELOAD peers. Each peer is identified by a Node-ID that is generally assigned by enrollment server as part of the overlay joining process. The transport component defines several messages types that will be utilized by the different Usages for data storage and resource discovery.

The storage component is responsible to process data storage and message retrieval. It is in charge to verify that an incoming data value is correctly stored by this peer and to respond a delivery status by applying the transport component. On receiving a storing request for a specific *Kind* by a another peer, it queries the corresponding Usage whether the requester is

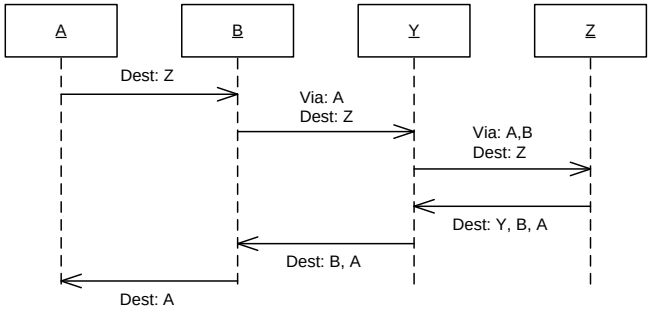

Figure 4: Symmetric and recursive routing procedure

allowed to store at this peer and also queries the topology plugin whether this peer is responsible for that Resource-ID. Additionally, the storage component is responsible to store replicas for the given data values at other RELOAD peers.

The RELOAD protocol is designed to be independent from the overlay routing algorithm. This is achieved by the topology plugin that abstracts the overlay routing from other RELOAD components. This enables to use any kind of structured or unstructured overlay algorithms that define the Routing Tables for message forwarding. The used topology algorithm also defines how for a given resource the corresponding Resource-ID is formed and defines the address range for which a peer is responsible.

Network Layer At the network layer the Forwarding and Link Management Layer is responsible to pass messages to next peers. This component establishes and maintains the network connections to other peers (while traversing NATs and firewalls using ICE [5]) as required by the topology plugin. Additionally, the forwarding component is responsible for setting message framing headers used to provide message reliability, congestion control and flow control. The default message delivery is symmetric recursive routing as shown in figure 4. A peer *A* sending a message to a peer *Z* retrieves the next hop *B* by looking up its routing table for destination *Z*. The *B* repeats this procedure and retrieves *Y* as next hop and inserts *A* into the via list. *Y* knows a direct route to *Z* and sends the message to *Z* and adds *B* into the via list. The response from *Z* to *A* then follows the entries in the via list beginning with *Y*.

Link Layer At this layer the protocol specification actually ends, since RELOAD does not define a new protocol at (real) transport layer. Instead of this, RELOAD referrers to the usage of secure transport protocols like Transport Layer Security version 1.2 (TLS) [8] and Datagram Transport Layer Security (DTLS) [9].

3.2 Defining Usages and Kinds

One of the most significant features of RELOAD originates from the possibility to serve as P2P data storage and connection service for a variety of different applications. Overlay networks commonly use some hash algorithms (e.g. SHA1 [10]) to calculate for a given data value an overlay ID – the storage position in a flat overlay address space. To allow storage of multiple application specific data values, the RELOAD base protocol introduces the *Kind* concept. Multiple different *Kinds* of data values can be stored under a common *Resource-ID*. Each Kind can belong to one or more application specific Usages, e.g., a SIP Usage could store a registration mapping Kind and a Kind for voice-mail recording. Additionally, a Usage need to define how the *Resource Name* for a data is formed. The hashed Resource Name (using the hash function from the topology plugin) then results as the Resource-ID.

Each Kind is identified by an own Kind-ID, a numeric code point that is assigned by the IANA [11]. The RELOAD specification defines three different data models for Kinds:

Single value: Stores a single data value.

Array: : Stores a list of single values with a numerical index.

Dictionary: Stores a list of single values using a specific key indexing each data entry.

Data structures types are not restricted to these three structures, but however, every Usage must define a data model which is used for storage.

RELOAD Usages need to define which access control policies are used for each of their Kinds. Access control policies define under which circumstances a peer is allowed to write or modify a data value in the overlay. The RELOAD security system is based on a central enrollment server that assigns each overlay peer a digital certificate that includes a peer *user name* (e.g. a SIP address) and the *Node-ID*. Therefore, the actually defined access control policies are derived from those identifiers in the certificate. For instance in a USER-MATCH policy, a peer is allowed to write a data at a specific Resource-ID, if the store request is signed with a key that is associated with a certificate whose user name hashes (using the hash function for the overlay) to the Resource-ID for the resource.

RELOAD is also designed to establish direct transport connections between peers that will be used by the applications. RELAOD therefore provides the *AppAttach* request. This overlay message exchanges contact informations (IPv4/IPv6 addresses), proposes candidates for NAT traversal (if needed) and defines the application that requests a transport connection. The latter definition is made by setting the 'port' value in the AppAttach request to the corresponding port of the desired application (e.g. setting port=5060 indicating SIP).

4 Application Usages

The RELOAD specification just included a very basic set of Usages to enable services that are needed for a reliable and trustworthy overlay network. The default way of defining a Usage is to specify an extra Internet draft that includes all specification described in section 3.2. The following section presents three of the first Usages for the RELOAD base protocol.

4.1 TURN Server Usage

The TURN Server Usage is defined within the RELOAD base document and allows overlay peers to announce them selfs as possible TURN servers [12] utilized by other peers for NAT traversal.

Advertise TURN server ability

When a peer initially joins a RELOAD instance, it enters the so called bootstrap procedure where a peer needs to establish direct connections to join the overlay maintenance (e.g. populate it's own routing tables). During this process it will notice by ICE checks [5] whether its IP address belongs to the public or private address range – if it is behind a NAT or not. If not, this peer is a candidate to become a TURN server and should advertise itself by following the algorithm shown in 1.

```
1 turnDensity = tD #a small integer
2
3 if(IP==public):
4   for d in range('1', turnDensity)
5     store(TurnServerKind,(hash(Node-ID+d)))
```

Listing 1: Advertising TURN Server ability

As the result of performing this algorithm, the TURN candidate publishes itself at various 'randomly' selected overlay addresses. The *turnDensity* value thereby must be adjusted to the size of the RELOAD overlay finding a appropriate ratio between advertisement and Spam.

Finding a TURN server

In the case that a peer is behind a NAT and needs to find a TURN server for establishing a transport connection for some reason, it follows the algorithms shown in code snippet 2.

```
1 hasTURNserver = 0
2 while hasTURNserver == 0:
3   # Queries the responsible peer for the random Resource-ID, if it
4   # has stored a mapping to a TURN server enabled peer
5   Resource-ID = random.randrange(1, 2^160) # default address range
6   hasTURNserver = find(TURNServerKind,(random (Resource-ID) ) )
7 fetch(TURNServerKind, Resource-ID) # retrieves contact to TURN server
```

Listing 2: Finding a TURN server

A searching peer request randomly selected overlay addresses queering the receiver if it has stored any *TURNserver* Kinds. This is achieved using the RELOAD *find* request message which will be answered with 1 if the requested peer has stored a TURNserver Kind, else 0. On a confirmative response, the searching peer then performs a RELOAD *fetch* request to the owner of data and retrieves the TURNserver Kind – the contact information to the TURN server enabled peer.

4.2 SIP Usage

The initial goal of RELOAD was to define a P2P signaling protocol used to support multimedia conferencing with the Session Initiation Protocol, a P2PSIP protocol. Accordingly, the first Usage that has been defined as extra an Internet draft was a SIP Usage for RELOAD [13].

Registering a SIP URI

The first task for a SIP Usage is to replace the traditional SIP proxy/registrar architecture in SIP. The registration of an end user device is performed by storing mappings for an Address-of-Records (AoR) to an overlay address, e.g., sip:alice@dht.exmaple.com->e4f3a. Address are registered by sending a RELOAD *store* request to the Resource-ID, which is the result by hashing the desired AoR. The SIP Usage allows two kinds of SIP registrations:

- Mapping from AoRs to destination lists (a list of overlay IDs).
- Mapping from AoRs to other AoRs.

The first case is stored together with a *contact prefs* denoting an preferred order under which a user wants to be contacted. For example, a user could set the Node-ID that registered for the user's office as first preference and afterwards its home Node-ID. The second mapping scheme can be used to delegate calls to other AoRs. For instance, if a CEO want to forward

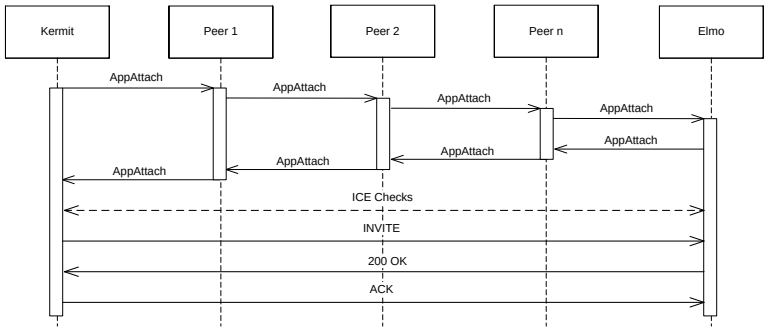

Figure 5: Establishing a SIP session using RELOAD

all incoming office calls to its secretary he could set a mapping 'sip:ceo@dht.example.com' -> 'sip:secretary@dht.example.com'.

A user that wants to resolve an AoR, just hashes the AoR and performs a *fetch* request as shown in 3. The request will be routed throughout the overlay to the owner of the address mapping. If the answer contains a destination list, then one of the retrieved Node-IDs is used to establish a SIP session as described in 4.2. If the answer contains another AoR that is not a Globally unique Routable User agent URI (GRUU), then the calling party must repeat this procedure until it retrieve a destination list or a GRUU. By retrieving a GRUU, the calling party must establish a SIP session using the traditional proxy/registrar way.

```
1 fetch(SIPKind,(hash('sip:elmo@muppets.com')))
```

Listing 3: Resolving a SIP URI

Establishing a SIP session

Th next step after resolving the AoR is to establish a SIP session to the remote end-system. The SIP Usage therefore defines a mechanism how to apply the AppAttach request as shown in figure 5. As any other overlay message, AppAttach is routed throughout the overlay from the caller to the callee. It thereby carries the IP addresses of both parties, indicate a SIP session setting port to 5060 and already carries some ICE candidates (e.g. possible TURN or STUDN servers). The callee responds by also sending an AppAttach carrying its own contact informations. Once the contacts are exchanged, a direct transport connection can be established and ICE checks are made to determine if and which hosts will be used for NAT traversal. Finally, an ordinary SIP session (INVITE->OK->ACK) will be established using the already created transport connection.

4.3 Service Discovery Usage

The possibility of RELOAD to support many different applications allows providers to publish their services (e.g. a search engine) into the P2P overlay. A naive solution would be to register a well known Resource Name (e.g., Google) and storing a mapping from a Resource-ID to the IP address of the service provider. This approach has particular disadvantages if the provided service is high frequently asked. Since all incoming service requests will be answered by one single peer, it can run into a scaling problem. Depending on the service been published, a single peer may store an amount of data or a huge number of service requests could overload the storing peer.

The Service Discovery Usage for RELOAD is a proposal to distribute service lookups onto several peers, that are arranged in a tree structure. This Internet draft is consequently porting the Recursive Distributed Rendezvous (ReDir) principle used in OpenDHT [14] to RELOAD.

Publishing a Service in RELOAD

In the Service Discovery Usage [15], each service is assigned by an service-specific identifier called namespace. The service providers are arranged in a tree structure, one ReDir tree per service. Each node in the ReDir tree is associated with a tree position (i, j), with i is the tree level and j is the $j-th$ position in a level counted from the left. The overlay address that is associated with a peer that stores mappings for service providers for a particular namespace is a $hash(NAMESPACE, i, j)$.

The ReDir registration follows the algorithm shown in 4. Beginning with some default tree depth value *Lstart* a peer p with a Node-ID *p.node-ID* registers a mapping using a *store* request. It is sent to a peer that is uniquely identified by result of *I(l,p.node-ID)*, which is an interval at level l in the ReDir tree that encloses $p.node-ID$. P then fetches all stored data at $I(l, p.node-ID)$ in order to determine if *p.node-ID* is the lowest or highest Node-ID in this interval. If true, the registration proceeds tree up and down (setting *Lstart* +1 and -1) until the previous condition is false, or if p becomes root of the tree, respectively becomes the only leaf in a sub-tree. If registration is complete, the last *Lstart* value is cached as new $Lstart$ for further service registrations.

The result of this registration scheme is a ReDir tree where each service provider may has multiple registrations, one at each level of the tree. From a tree position (i, j), a peer can derive an interval within the entire overlay address space for which it is responsible by using formula 1:

$$[2^{160} * b^{-i} * (j + (b'/b), 2^{160} * b^{-i} * (j + (b' + 1/b)], \quad with \quad 0 <= b' < b \tag{1}$$

```
1  def reDirRegister(Lstart):
2    Lstart = 2 # default 2
3    myNodeId = dht.getNodeId()
4    resultSet[] # List of Node-IDs of other service providers
5    # I(l,k) unique interval at level l in the ReDir tree that encloses k
6    store(ReDirKind,'NAMESPACE',I(Lstart, myNodeId))
7    resultSet = fetch(ReDirKind,'NAMESPACE',I(Lstart, myNodeId))
8    # size==0, myNodeId is the only at this leaf
9    # Lstart==0 this peer is root of tree
10   if resultSet.size() == 0 || Lstart == 0:
11     return Lstart # Lstart for next Registration
12   for i in range (resultSet.size()):
13     if(myNodeId < resultSet[i] || myNodeId > resultSet[i]):
14       # walking recursily tree up and down
15       walkUpTree   = Lstart--
16       reDirRegister(walkUpTree)
17       walkTreeDown = Lstart++
18       reDirRegister(walkTreeDown)
19     else:
20       return Lstart # Lstart for next Registration
```

Listing 4: Registering as service provider

For instance, the root peer (i=0) of the tree is responsible for the entire address space. Peers at level 1 are responsible for each a half of the address space; at level 2 each peer is responsible for a quarter of the address space and so forth.

Discovering a Service

A service lookup is analogue to the service registration without store request. A peer p sends a *fetch* request towards a peer that is storing ReDir registrations for an interval *I(level, p.node-Id)* (starting with *Lstart=2*). If *p.node-ID* is greater or smaller than every other Node-ID in the result set, p repeats service lookup with setting *level=level-1*. If the result set contains a $Node-ID > p.node-ID$ and there is no other Node-ID in between, then p found its succeeding service provider. Else, p repeats the service lookup setting *level=level+1*. The last level value will be cached as new *Lstart* for further service lookups.

The load of storing ReDir Kinds using Service Discovery Usage, is uniquely distributed onto all peers in the tree. After initial service lookups are performed, which are already distributed onto four ReDir nodes, further service requests will only contact ReDir nodes that are in a same address range as the requesting peer.

5 Conclusion and Outlook

The RELOAD base protocol facilitates end-to-end communication in a P2P system by providing an abstract storage and message transport service to its peers. Security is assured through an central enrollment server that provides credentials for each overlay peer, thus each operation can be authenticated. RELOAD includes mechanisms for NAT and firewall traversal by using ICE protocol in combination with overlay peers that announce their TURN server ability. Message routing is independent from underlying distributed hash algorithm, since a generic topology plugin provides an abstract interface to other components. Network security is guaranteed by using secure transport protocols. RELOAD introduces the Usages concept that allows a variety of different applications to utilized RELOAD as service platform. Usages therefore define their specifications (Kind data structure, Access control policy, ...) within an external Internet draft.

This document presented three applications that defined Usages for RELOAD. The TURN Usage allows peers to establish transport connections although they are located behind a NAT. Since RELOAD was designed as a P2PSIP, the definition of a SIP Usage consequently followed the RELOAD specification. It allows the establishment of SIP sessions without the burden of the traditional proxy/register architecture in IP telephony. A mechanism to publish services in a scalable manner is given by the Usage for Service Discovery. Service providers store their mappings at several overlay peers that ordered in a tree structure. Clients search along the tree structure to find their responsible service provider.

These characteristics of the RELOAD base protocol makes it significant for the distributed conferencing approach [3]. Firstly published in [16] and presented as proposal for a new RELOAD Usage for Distributed Conference Control (DisCo) in [4], this work is still in progress. The area of media parameter negotiation in distributed conferences and a secure access control policy for shared resources in RELOAD are examples for future work.

References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," IETF, RFC 3261, June 2002.

[2] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol," IETF, Internet-Draft – work in progress 10, August 2010.

[3] "Infrastructure Independent Conferencing," http://users.informatik.haw-hamburg.de/ ubicomp/projekte/master09-10-aw1/knauf/bericht.pdf, 2010.

[4] A. Knauf, G. Hege, T. Schmidt, and M. Waehlisch, "A RELOAD Usage for Distributed Conference Control (DisCo)," IETF, Internet-Draft – work in progress 00, June 2010.

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

[6] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proc. of 7th conf. on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'01)*. New York, NY, USA: ACM Press, 2001, pp. 149–160.

[7] H. Zimmermann, "OSI Reference Model — The ISO Model of Architecture for Open Systems Interconnection," *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425–432, April 1980.

[8] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," IETF, RFC 5246, August 2008.

[9] E. Rescorla and N. Modadugu, "Datagram Transport Layer Security," IETF, RFC 4347, April 2006.

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

[11] A. Mankin, S. Bradner, R. Mahy, D. Willis, J. Ott, and B. Rosen, "Change Process for the Session Initiation Protocol (SIP)," IETF, RFC 3427, December 2002.

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

[13] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, "A SIP Usage for RELOAD," IETF, Internet-Draft – work in progress 05, July 2010.

[14] S. C. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu, "OpenDHT: a public DHT service and its uses." in *Proc. of 8th conf. on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'02)*, R. Guérin, R. Govindan, and G. Minshall, Eds. ACM, 2005, pp. 73–84.

[15] J. Maenpaa and G. Camarillo, "Service Discovery Usage for REsource LOcation And Discovery (RELOAD)," IETF, Internet-Draft – work in progress 00, October 2009.

[16] A. Knauf, G. Hege, T. C. Schmidt, and M. Wählisch, "A Virtual and Distributed Control Layer with Proximity Awareness for Group Conferencing in P2PSIP," in *Proc. of 4th conf. on Principles, Systems and Applications of IP Telecommunications (IPTComm'10)*, ser. Digital Library. New York: ACM, August 2010, accepted for publication.

List of Figures

Listings