Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Seminar Ringvorlesung - Wise 2010/11

Alexander Knauf

A RELOAD Usage for Distributed Conference Control

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Contents

1 Introduction

The ubiquitous availability of broadband Internet connections and the constantly increasing computing power of the consumer devices, enable the use of decentralized peer-to-peer networks which often reach a better scalability than traditional Client/Server architectures. Since the early 2000s many so called structured overlay networks, e.g. Chord, Pastry and CAN [1, 2, 3], have been developed and are proved in their functionality to provide services for millions of clients while featuring adequate response times. The advantages of P2P overlays motivated re-innovations of P2P solutions for problems in centralized networks. Following this principle, the International Engineering Task Force (IETF) is developing an Internet standard for P2P networks that provides an abstract communication layer to be used by a variety of applications called RELOAD [4]. The REsource Location And Discovery protocol was initially designed to decentralize the existing Proxy/Registrar infrastructure that is needed by the Session Initiation Protocol (SIP) to establish end-to-end connectivity for multimedia communication. However, the RELOAD base protocol allows many other applications to specify their requirements in RELOAD *Usages* in order to utilize RELOAD as communication layer.

This document presents the problem statement and objectives for a new RELOAD Usage for distributed conference control which is proposed for my master thesis. Most solutions for multimedia conferencing use a central server architecture to interconnect the participants of a multiparty session. The central server is the locator of conference, several services and gathers and distributes all media streams [5]. The resulting star topology in traditional conferencing architectures has an $O(N)$ complexity at the server and limits the scalability of large conferences. Hence, a P2P and decentralized solution might be able to provide a stable conferencing service in which the load for maintaining multimedia streams is unified distributed among the participants. Using RELOAD base protocol as communication layer facilitates register and lookup functions and provides a trustworthy connection setup. In Distributed conference control, the role of the central server is divided onto several individual end user devices each responsible for only a subset of participants of the entire conference. Minimizing signaling and media delay and jitter, participants of a distributed conference select the topologically closest end user device to join the conference. This document also presents a proposal for a secure mechanism that allows conference controllers to request participants to become a manager of a conference to enable load balancing.

The reminder of this document is structured as following. Section 2 introduces into traditional conferencing architectures and explains the resulting problem statements. It furthermore presents the proposals and approaches for distributed conference control using an RELOAD overlay. Section 3 gives an overview on preliminary works for the master thesis explains the starting situation. The proposed architecture and used technologies are presented in section 4 and section 5 takes a look at possible risks while developing the master thesis. Finally, section 6 concludes at gives look on future work.

2 Problem Statement and Objectives

2.1 Centralized Conferencing Architectures

The default architecture for multiparty voice and video conferences uses a central server that interconnects all participating end user devices [5, 6]. Conference participants join a multiparty session by sending invites requests using a signaling protocol, e.g. SIP or H.323 [7, 8], to the conference server. Those request commonly carry contact informations (IP, port), user credentials and the supported voice and video codecs or even device capacity informations. The content is encapsulated in a session description protocol (e.g. SDP [9, 10]). A conference server has typical characteristics. It is most often accessible by a globally routeable Unified Resource Identifier (URI), is located at a dedicated server infrastructure and provides a powerful Multipoint Control Unit (MCU) that is able to gather and distribute multiple multimedia streams. Additionally, conference servers commonly provide functionalities for conference notification services [11] and floor control [12]. Notifications services convey the state (e.g., list of participants, used media) of a conference to its participant, while floor control enables the management of the conference parameter (e.g., media setup, conference policy).

These rich functionalities and high hardware requirements for a conferencing environment demand a suitable server infrastructure. Hence, carriers of such an infrastructure charge a fee for the usage of conferencing service. The migration of conferencing services from servers onto a common end user devices is limited by its processing power and its global accessibility without external assistance. A decentralized and peer-to-peer solution must face those challenges.

2.2 Distributed Conference Control

The Session Initiation Protocol (SIP) [7] is the most widespread signaling standard for IP telephony and multimedia/multiparty conferencing. It is an application layer protocol for creating, modifying and terminating sessions between one or more participants. Multiparty sessions are realized in a centralized architecture called *tightly could model* [5] that uses a single contact point called *focus* of a conference. The *focus* locates the multiparty conference and identifies it by a unique routable URI– the conference URI or ID. It listens on SIP invitation requests, negotiates media parameter with each calling party and controls media serves and notification services.

The distributed conferencing architecture [13] remains a in tightly coupled singling model, but uses multiple independent entities that perform precisely the tasks of an conference *focus*. Each of these *focus peers* performs conference management and media distribution for only a subset of the connected participants. This architecture reduces the computation effort at

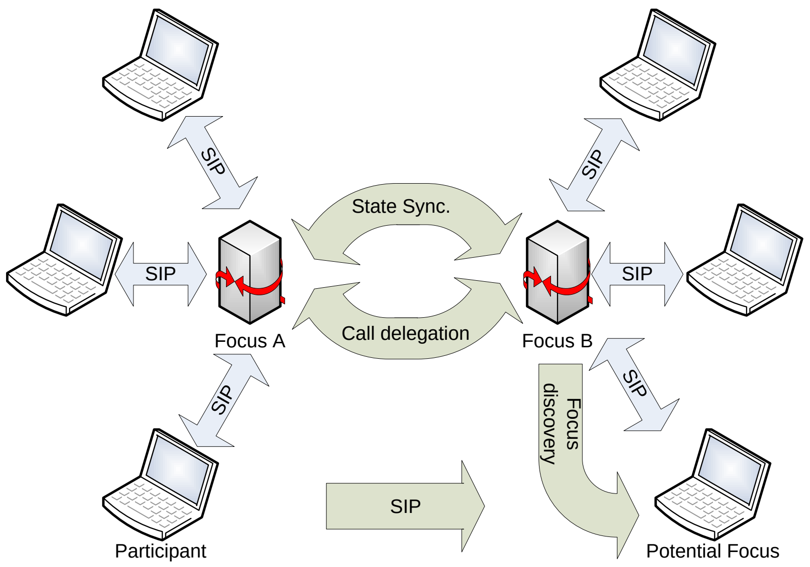

Figure 1: Distributed conference control scenario

a single device by distributing the load onto multiple systems. The *focus peers* are interconnected in order to synchronize the global conference state and to upstream the media of their participants. As shown in figure 1 each *focus peer* maintains its own signaling relations to its connected participants. State synchronization takes place between Focus A and Focus B using SIP specific event notification mechanism [14]. SIP Notify messages are carrying an XML based event package for distributed conferences [15] that is designed to convey information about roles and relations of the conference participants. *Focus Peers* obtain a global view on the conference state and use this information for load balancing operations. The latter are performed by *call delegation* and *focus discovery* mechanisms. *Call delegation* is performed if a conference controller reaches its threshold in serving new participants. It then delegates an incoming call to a remote focus peer using a SIP REFER request [16]. *Focus discovery* is performed if the conference as whole reaches its limit to serve new participants. An overloaded focus peer then requests participants to become a new conference controller handling new joining requests.

The distributed conferencing scheme is designed for downwards compatibility for SIP implementations that are unaware of distributed conferencing architecture or even a of multiparty scenario. This is achieved by a standard conform *record-route* header mechanism [13] performed by the *focus peers* in a conference. Shown in listing 1 is a conference focus that invites a user 'Bob' to participate the multiparty conversation identified by the URI *sip:conference@dht.example.com* (see lines 4,7). The request seems to be a standard invitation of a focus as indicated by the *isfocus* tag in the contact header which would be just

```
INVITE sip:bob@dht.example.com SIP/2.0
Call-ID: 0815@141.22.26.55
CSeq: 1 INVITE
From: <sip:conference@dht.example.com>;tag=134652
To: <sip:bob@dht.example.com>;tag=643684
...
Contact: <sip:conference@dht.example.com>;isfocus
Record-Route: <sip:alice@dht.example.com>
...
```

Listing 1: SIP INVITE request unsing record-route mechanism

ignored by conference unaware applications. Striking is the additional *record-route* header that refers to a SIP user 'Alice'. Actually, Alice is associated to a group focus peers that control this multiparty session as a distributed conference. The record-route header effects that further signaling messages by 'Bob' will preliminary be routed to 'Alice'. Since she is aware of the distributed signaling and is the responsible focus peer for 'Bob', she intercepts those messages and replies them in the role of the conference focus. For 'Bob', the record-route is just a standard extension header that prescribes him to add 'Alice' into the *route-set* [7] for further signaling messages.

2.3 Introducing RELOAD-base Specification

In ordinary SIP, a client application registers its Address-of-Record (AoR) at a dedicated registrar server [7]. Commonly, a registration of an AoR is bound to a contract with a server provider, thus a client need to authorize itself by sending its credentials (e.g., user name, password). The AoR of the SIP user agent then is bound to a domain name (like sip:alice@provider.com), that can be resolved to the IP address using DNS lookups. SIP sessions between two SIP user agents then can be established using SIP proxy servers. The *outbound proxy* of the calling party therefore resolves the domain name of the called party and forwards or redirects *callee* to the *inbound proxy* of the called party. The inbound proxy then resolve the host part of the called SIP URI to its IP Address and forwards or redirects the invite request. By retrieving the IP address of the called party, a direct SIP session can be established. This proxy/registrar architecture is called *SIP trapezoid* [7].

Focusing on a P2P scenario, neither the calling party nor the called party can fall back on a proxy/registrar infrastructure to resolve URIs or to establish SIP session via a SIP trapezoid.

A solution to resolve SIP records and to establish sessions is presented by the P2P signaling protocol RELOAD [4]. The REsource Location And Discovery protocol provides a generic

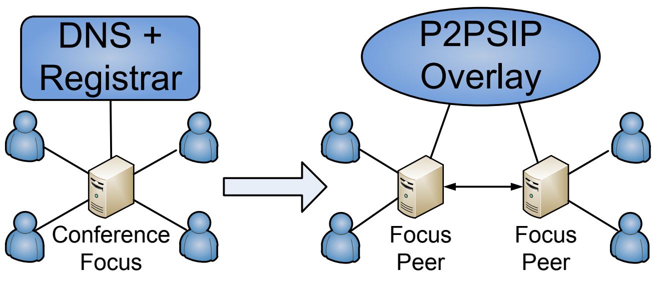

Figure 2: Vision of distributed conference control

storage and messaging service that is formed by a cooperative network of peers in an overlay network. Peers of a RELOAD overlay can store values on remote hosts which are available for for all client applications of an overlay instance. In addition, RELOAD provides the functionality to establish direct transport connections between hosts, including traversal of NATs and firewalls. RELOAD is designed as an abstract communication layer that can be used for a variety of different applications. These have to specify their protocol requirements in so called *Usages*. Each *Usage* therefore has to specify the following information:

- Definition of data structures that will be stored at the overlay. In RELOAD, those data structures are called *Kinds*.
- Definition and registration of *Kind-IDs* for any Kinds of the Usage.
- Definition of access control rules to the specified *Kinds*.
- Description how values will be merged after network partition.
- Definition how the name of a data value is formed. The name of a value implies where a data will be stored at the overlay, since the hash over the name defines its storage location.

The rich functionalities of the RELOAD base specification and its extensibility for applications by just defining new *Usages* motivated to the definition of a RELOAD Usage for distributed conference control as proposed in section 2.2. As shown in figure 2 the RELOAD overlay replaces the traditional SIP proxy/registrar infrastructure and provides *focus peers* a platform for P2P multiparty conference environment.

2.4 Usage for Distributed Conference Control

The RELOAD Usage for distributed conference control provides members of an overlay instance to register a variable conference URI that identifies a single multiparty session that is

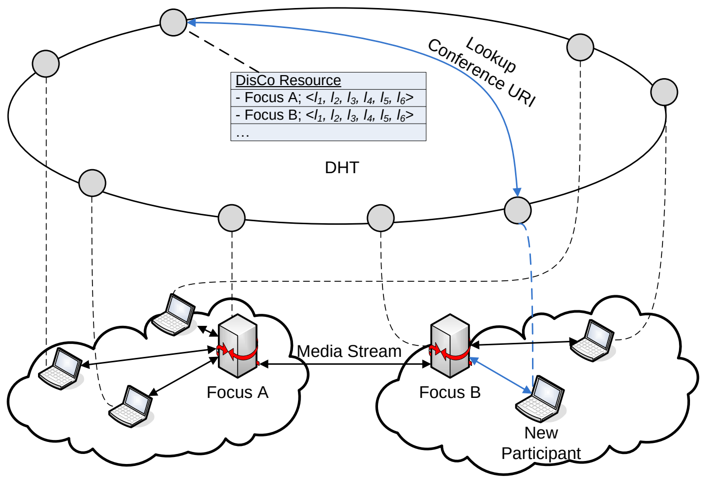
Lookup
Conference URI
DisCo Resource
- Focus A; $<l_1, l_2, l_3, l_4, l_5, l_6>$
- Focus B; $<l_1, l_2, l_3, l_4, l_5, l_6>$
…
DHT
Media Stream
Focus A
Focus B
New Participant

Figure 3: Reference scenario for distributed conferencing using RELOAD

located by several *focus peers*. The corresponding RELOAD Kind data structure called *DisCo-Registration* stores mappings from the conference URI to the overlay addresses of the focus peers– to their *Node-IDs*. Additionally, each DisCo-Registration stores a topological descriptor called *coordinate* value that announces an relative position in the network of an end user. It is used to optimize the interconnection graph among the conference participant and its responsible focus peers. A reference scenario is show in figure 3.

A RELOAD peer that intents to participate a multiparty conversation uses RELOADs lookup functionality to resolve the conference URI. It retrieves a dictionary that contains all actively managing focus peers and all registered but not actively managing focus peers. The joining then chooses the relative closest focus peer to join the conference using on the *coordinate* value. It establishes a transport connection using RELOAD functionalities upon it creates a SIP session to participate the distributed conference as presented in 2.2. The focus peer announces the arrival of new participant and connects it with the media streams of the conference.

Participants of a conference are obliged to put their device capabilities at the disposal for the conference to enable a constant quality of service while having an increasing multiparty session. Hence, participants receive the permission to register their mappings into the DisCo-Registration by the creator of the conference or other already authorized focus peers. However, a shared write access of several independent RELOAD users on a single overlay location is not proposed in the RELOAD base specification [4]. Each RELOAD is issued an X.509 certificate [17] by an enrollment server that represents the central authority of a RELOAD instance. The certificate contains, inter alia, a 'user name' value and the 'Node-ID' of each RELOAD user.

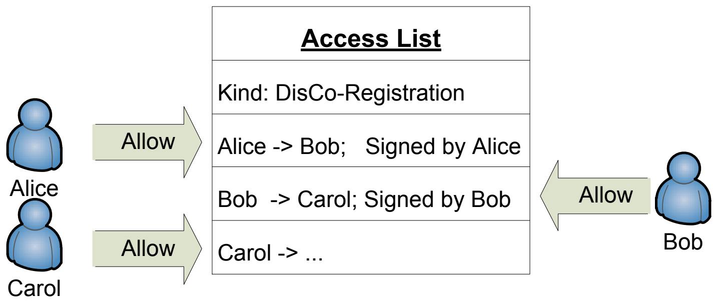

Figure 4: Example of an Access List

It allows a user write access to those overlay addresses that are equal to the hash over the user name or equal to the hash over the Node-ID. Unauthorized request will be declined. The RELOAD base specification, however, provides the ability to define new access control policies [4].

This motivated the proposal for new Usage for Shared Resources (ShaRe) [1]. It provides a mechanism that enables the owner of the certificate that allows write access at a specific location, to delegate the authorization to write a specific Resource/Kind pair (the *shared resource*) to other RELOAD users. Therefore, the owner of an existing overlay resource stores another RELOAD Kind called *Access List*. As shown in figure 4, the resource owner 'Alice' delegates the permission to write a shared resource to the RELOAD user 'Bob'. Bob in turn, further allow 'Carol' write access to the resource and so far. The corresponding access control rule then permits write access to shared resource to those users that are registered in the access list to a specific resource. Furthermore, by having all trust delegation in a 'from'/'to' relation, each list item can be traced back to the initial access list entry written by the resource owner. Thus, it serves as the trust anchor for the entire trust delegation tree. Looking back to distributed conferencing, the initiator of a conference than can create an Access List to share the DisCo-Registration and allow focus peers to store their mapping into it.

3 Starting Situation

3.1 Stable Concept

The proposal for a distributed conference control that utilizes the RELOAD base protocol has been modified quite a lot during the last year. The first concept for the 'DisCo' Usage was

[1]This reference was not yet published at release of this document

presented at the 78th IETF conference in Maastricht. The received feedback was very fruitful but included a rewrite for many proposed concepts. For instance, the first DisCo approach provided a more intensive usage of the central enrollment server. The initiator a of conference was meant to request a new digital certificate for each new conference it creates. The corresponding feedback was, that the enrollment server should only be issued on enrollment of a RELOAD user.

The second revised draft for distributed conferencing used self generated certificates which are created by the conference creator. They should serve as a shared secret that allows focus peers to write the DisCo-Registration. Since this approach seemed to be not elegant, the next draft version will include the *shared resource* using access lists for a cooperative write access. This approach will be presented in the upcoming 80th IETF conference in Praque.

3.2 ID/Locator Separation in SIP

The protocol scheme for distributed conferences was part of my bachelor thesis [18]. The presented *record-route* mechanism that allows a distribution of the conference ID to several focus peers as locators, was implemented and evaluated with a Java SIP stack. The results showed that the logic of the SIP stack was not influenced of the ID/Locator split using the record-route technique. Further, the evaluation showed that the signaling delay remains constant while an increasing conference.

3.3 Event Package for Distributed Conferences

State synchronization of focus peers in a distributed conference requires a common protocol that conveys the local state at each controller. As part of the specification for the DisCo Usage, a XML based event package for distributed conferences was developed. It provides sufficiently information about roles and relation among the conference participants and takes possible race conditions by concurrent event change into consideration.

3.4 RELOAD Stack implementation

A main problem while concept and development phase was that there are no open RELOAD implementations available. The RELOAD protocol definition is still in proceeding and it is not clearly figured out whether the RELOAD will settle down as a standard overlay protocol. On the opposite, a self-implementation of RELOAD would have been too complex for a single developer.

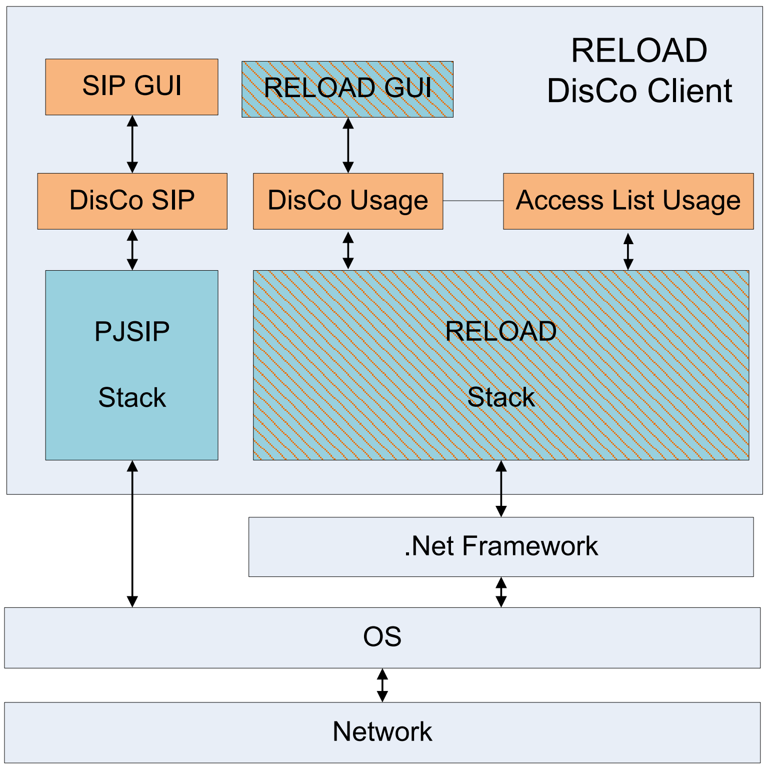

Figure 5: Project Architecture

Finally, I have obtained a RELOAD implementation by a cooperation with a popular Internet provider. The stack provides almost every functionality of the base protocol and also implements the SIP Usage for RELOAD [19].

4 Architecture and Technologies

4.1 Thesis Project Architecture

The implementation of proposed conferencing solution will to be written in the C-Sharp and C/C++ programming language. Mainly reasoned is this decision by the RELOAD and SIP stack implementations that are written in C/C++/C#. An overview of the entire software architecture is given by figure 5. The orange colored boxes display those components that will be completely new developed. The blue components show existing components that will just be used as library and hatched boxes refer to components that exist or need to be modified.

The PJSIP stack [20] is an open source SIP stack that supports several SIP extensions and features. It provides the full SIP core functions [7] and implements many SIP and SDP [10] extensions like SIP specific event notification mechanism [14] which will be used for conference state synchronization. The API is easy to use and provides many ports to other operation systems.

The utilized RELOAD stack was implemented within the scope of a feasibility study by an popular Internet provider [2]. It was designed to show how a P2P overlay support IP telephony on mobile devices. Therefore, the entire stack and IP telephony functions can be deployed onto mobile devices running Window Mobil 6. Additionally, the stack provides a visualization mechanism based on Google Maps. It displays all entities and devices connected to a RELOAD overlay and reports their status. However, several RELOAD specification were out of development scope and demand a new or re-implementation.

5 Risks

5.1 Thesis compatible to Internet Standards

The scope of my master thesis is to define a new Internet draft suggested as Internet standard. Several mechanisms and protocol schemes might be in conflict with standard behavior. Conflicts can require conceptual reissues that are followed by re-implementations of the thesis project. Only a meticulously elaboration of approaches and continuous feedback by third persons can counteract a conceptual wild growth.

5.2 Evaluation of Proximity-awareness

A partial aspect of distributed conferencing provides the possibility to select the relatively closest focus peer on joining a distributed conference. This aspect is targeted to be evaluated using the PlanetLab [21] which provides a platform for development and deployment of distributed application in a global scale. PlanetLab is a cooperation of several universities and research facilities around the world that provide a network of a hundred severs running Linux OS.

The latter fact might be a obstacle since most of the proposed application is running upon the .Net framework. The cross platform Mono might be a solution, but it remains the perspective that an evaluation could be very difficult.

[2]For reasons of confidentiality, our industrial partner will not be named publicly.

6 Conclusion and Outlook

The concept for a RELOAD Usage for distributed conferencing seem to be resilient and shows an approach for a decentralized P2P solution for multimedia conferencing. The SIP protocol scheme to separate the conference identifier onto multiple locations works transparently to those SIP implementation that are not aware of distributed conferencing environment. The definition of an XML based event package for distributed conference allows the controllers of conference to synchronize the entire conference state. Mechanisms like call delegation and focus discovery provide a better scalability for increasing conferences compared with traditional centralized solutions. The Usage of a RELOAD overlay to register a conference URI and establish direct connections goes in accordance with the intension to be independent from dedicated server infrastructure. The second Usage for shared resources is not limited for usage of distributed conferencing. Rather, it provide a variety of other distributed applications the possibility to cooperatively share specific resources.

The preliminaries are set to start with the master thesis. Given are an extensive discussed concept and the previous implementations and evaluations during my studies. In particular, the possession of a RELOAD stack enables an implementation of the proposed concepts and was one of the most critical factors for a successful master project. A prototype implementation will demonstrate that P2P conferencing solution are possible and might be an alternative to traditional centralized conferencing services.

References

[1] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup service for internet applications,” in *SIGCOMM ’01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, (New York, NY, USA), pp. 149–160, ACM Press, 2001.

[2] S. Ratnasamy, M. Handley, R. M. Karp, and S. Shenker, “Application-Level Multicast Using Content-Addressable Networks,” in *Networked Group Communication, Third International COST264 Workshop, NGC 2001, London, UK, November 7-9, 2001, Proceedings* (J. Crowcroft and M. Hofmann, eds.), vol. 2233 of *LNCS*, (London, UK), pp. 14–29, Springer–Verlag, 2001.

[3] A. Rowstron and P. Druschel, “Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems,” in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, vol. 2218 of *LNCS*, (Berlin Heidelberg), pp. 329–350, Springer–Verlag, Nov. 2001.

[4] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, “REsource LOcation And Discovery (RELOAD),” Internet-Draft – work in progress 00, IETF, July 2008.

[5] J. Rosenberg, “A Framework for Conferencing with the Session Initiation Protocol (SIP),” RFC 4353, IETF, February 2006.

[6] M. Barnes, C. Boulton, and O. Levin, “A Framework for Centralized Conferencing,” RFC 5239, IETF, June 2008.

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

[8] ITU-T Recommendation H.323, “Infrastructure of audio-visual services - Systems and terminal equipment for audio-visual services: Packet-based multimedia communications systems,” tech. rep., ITU, 2000. Draft Version 4.

[9] M. Handley and V. Jacobson, “SDP: Session Description Protocol,” RFC 2327, IETF, April 1998.

[10] M. Handley, V. Jacobson, and C. Perkins, “SDP: Session Description Protocol,” RFC 4566, IETF, July 2006.

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

[12] G. Camarillo, J. Ott, and K. Drage, “The Binary Floor Control Protocol (BFCP),” RFC 4582, IETF, November 2006.

[13] A. Knauf, T. C. Schmidt, and M. Wählisch, “Scalable Distributed Conference Control in Heterogeneous Peer-to-Peer Scenarios with SIP,” in *Mobimedia ’09: Proc. of the 5th International ICST Mobile Multimedia Communications Conference*, ACM Digital Library, (Brussels, Belgium), pp. 1–5, ICST, Sept. 2009.

[14] A. Roach, “Session Initiation Protocol (SIP)-Specific Event Notification,” RFC 3265, IETF, June 2002.

[15] A. Knauf, G. Hege, T. Schmidt, and M. Waehlisch, “A RELOAD Usage for Distributed Conference Control (DisCo),” Internet-Draft – work in progress 01, IETF, December 2010.

[16] R. Sparks, “The Session Initiation Protocol (SIP) Refer Method,” RFC 3515, IETF, April 2003.

[17] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,” RFC 5280, IETF, May 2008.

[18] A. Knauf, “Scalable, distributed conference control in tightly coupled scenarios.” https://kataloge.uni-hamburg.de/DB=2/PPNSET?PPN=610041797, 2009.

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

[20] “PJSIP Stack.” http://www.pjsip.org/, 2007.

[21] “The PlanetLab homepage.” http://planet-lab.org/, 2010.

List of Figures

Listings