Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Anwendungen 1 - WiSe 2009/10

Alexander Knauf

Infrastructure Independent Conferencing

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Alexander Knauf

Thema der Ausarbeitung

Infrastrukturunabhängige Multimediakonferenzen

Stichworte

Peer-to-Peer, SIP, P2PSIP, Konferenzmanagement, Verteilte Systeme

Kurzzusammenfassung

Das Session Initiation Protocol (SIP) ist ein Signalisierungsprotokoll, welches zur Erstellung, Modifikation und Terminierung von Multimediasitzungen verwendet wird. Um den Standort des Kommunikationspartners zu ermitteln und die Mediendaten unter Konferenzteilnehmer zu verteilen, benötigt man eine dedizierte Serverinfrastruktur. In dieser Arbeit werden die Methoden und Abläufe bei der Erstellung von Multimediakonferenzen erklärt, eine Einführung in strukturierte P2P overlays gegeben und Alternativen zu der statischen Client-Server Architektur in SIP vorgeschlagen.

Alexander Knauf

Title of the paper

Infrastructure independent conferencing

Keywords

Peer-to-Peer, SIP, P2PSIP, Tightly coupled conferences, Distributed Systems

Abstract

The Session Initiation Protocol (SIP) is an application layer signaling protocol for creating, modifying and terminating multimedia sessions. To obtain the location of remote users and distributing media data among the conference participants, a dedicated server infrastructure is required. This paper gives an overview of how multimedia sessions with several participants are created. It further introduces into structured p2p overlays and proposes alternatives to the traditional client-server paradigm in SIP.

Contents

1 Introduction

Voice over IP (VoIP) and Voice and Video over IP (VVoIP) gain more importance since the capabilities of the end systems (CPU, Memory) and the connectivity to broadband Internet is increasing continuously. These techniques offer an alternative to traditional wired telephony for end users and telephone communications-providers, as well. Beside to point-to-point communication, VVoIP applications often offer the option to create multiparty conversations. Some of these conferencing applications are based on closed and proprietary protocols and limited in the number of participating clients. To provide multi-party video conferences, a dedicated media-mixer server called Multipoint Control Unit (MCU) [1] is required to distribute the media data among the participants. These server systems are expensive and need maintenance, thus service access may be costly. Because of the high complexity of processing multiple media streams and dispatching them to the end systems, these MCU systems are of limited scalability.

The Session Initiation Protocol (SIP) [2] is an open and standardized signaling protocol for creating, modifying and terminating multimedia sessions. It uses an infrastructure of SIP proxies and redirect servers, to interconnect end systems called *User Agents* that are addressed by SIP URIs. One model for conference with SIP uses a central point of control called *focus* [3] that is responsible for interconnecting and managing the multimedia session. The focus is located and identified by a conference specific URI, that must be globally unique and routable. It is often located on a conference server, that provides MCU-type abilities by distributing the media data among the participants.

The dependency of SIP on providers that offer proxy and redirect services motivated peer-to-peer based resource location and discovery approaches. Such solutions are built upon *distributed hash table* (DHT) algorithms, like Chord, Pastry or CAN [4, 5, 6]. DHTs provide scalable, flexible and robust lookup and routing functions. These approaches known as *P2PSIP* are work in progress and not standardized, yet. An open challenge in this domain is the area of highly scalable voice and video conferences. A dedicated conference server with MCU capabilities cannot be assumed in an P2P environment. Therefore, a distributed P2P-signaling protocol scheme based on SIP could serve for a distributed media mixing topology, and manage large multimedia conferences. Such distributed concept requires an abstraction of the conference URI that will be located by many peers instead of one conference focus in traditional SIP. These issues challenge for further research.

The remainder of this paper is structured as follows. An overview of traditional signaling and conferencing with SIP is given in section 2, and an introduction to peer-to-peer overlay networks in section 3. This paper presents an approach to a P2P-based virtual conference scheme in section 4 and conclusion and outlook for future work in section 5.

2 Conferencing with SIP

2.1 Signaling with SIP

The Session Initiation Protocol is an application layer signaling protocol for establishing sessions for point-to-point or multi-point communication. It is a text-based protocol with elements similar to SMTP and HTTP. SIP is independent of the underlying transport protocol (UDP, TCP) and can be used to carry session description information which allow communication end-points to agree on a set of compatible media types. The media negotiation is often done by sending Session Description Protocol (SDP) [7] data, informing the remote end-point about the supported media types. A typical call flow for initiating a SIP session called *Dialog* in shown in figure 1. The SIP INVITE request sent by the user agent *Piggy* (*caller*), initiates the dialog establishment and could look like below:

```
INVITE sip:kermit@sesamestreet.com SIP/2.0
Call-ID: 0815@141.22.26.6
CSeq: 1 INVITE
From: "Piggy" <sip:piggy@muppets.com>;tag=134652
To: "Kermit" <sip:kermit@sesamestreet.com>
Via: SIP/2.0/UDP 141.22.26.6:5060;branch=z9hG4bKf1
Max-Forwards: 70
Contact: <sip:piggy@141.22.26.6>
Content-Length: 159
...
```

A SIP request message starts (*request-line*), that defines the desired SIP *method* (here `INVITE`) and the requested *callee*. Communications end-points in SIP are called *User Agents* and are addressed by SIP Uniform Resource Locator (URI). A SIP URI starts with *sip:* followed by the composition of user name `@` the related domain or IP address (here *sip:kermit@sesamestreet.com*). After the request-line, a set of *header fields* are defining more information to establish a communication session described in RFC-3261 [2].

The `INVITE` request in figure 1, is firstly forwarded to the user agent's *outbound proxy* (muppets.com). A SIP *proxy* is a provider maintained signaling server, that is responsible to find the destination for the requested user agents. The outbound proxy informs the caller about the pending request processing by sending a 100 *trying* response message. The session initiation protocol uses DNS procedures to allow a client to resolve SIP URI into the IP address, port, and transport protocol of the desired callee. In this sample call flow, the request will be forwarded the requested user agent's *inbound proxy* (sesamestreet.com). As the callee is available, the end-point's SIP application or IP-telephone firstly responds by sending 180 *ringing* messages.

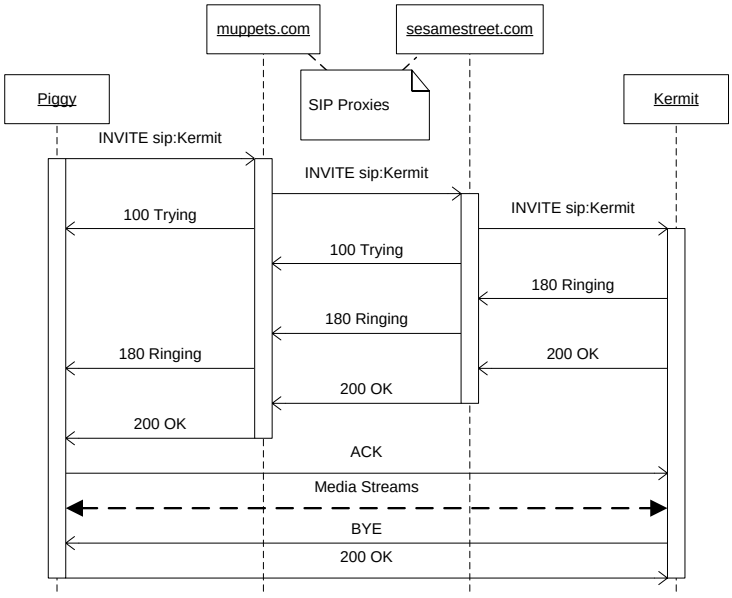

Figure 1: SIP dialog establishment via proxies

These will be interpreted as a ring tone at the callers side. If the callee answers the phone, it sends a confirmative 200 *OK* response message, thus the caller is informed of a successfully established call. From here, both communication end-points have negotiated the media parameters by exchanging SDP data and know the end systems locations from the data in the *contact* header field. Hence, the final ACK message can be send directly to the callee and the media session can be established. To end a communication session, a user agent sends a SIP `BYE` request receipted by a 200 *OK* response.

2.2 Traditional Conferencing

Using the definitions of the Internet Engineering Task Force (IETF) [8], there are three different models for multi-party communications. The so called *loosely coupled* model does not provide a signaling relation between the multi-party participants. Participation can be done by joining multicast groups and control information is learned from the used transport protocol (e.g. RTCP [9]). In a *fully distributed model* each participant manages a signaling dialog to all other remote participants in a full mesh. Finally, the *tightly coupled model* refers to a signaling relation between all participants and one central point of control, that negotiates media parameters to

establish voice, video or application sessions. In SIP, this central point of control is called the *focus* of a conference [10], displayed in figure 2.

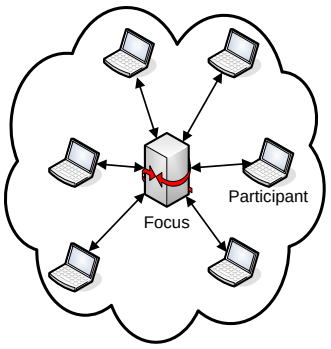

Figure 2: Tightly coupled conferencing model

It is identified and located by a conference specific SIP URI (Conf-ID) and must be globally unique routable. The former two models are not suitable for large scaling conferences, because of the complexity and may tedious signaling efforts and are out of scope of this paper. Regarding to the latter approach, such a conference specific URI needs to be created considering the domain name the SIP user agent is connected to. This can be done by querying a dedicated conferencing server to allocate and publish a conference URI (e.g. *sip:puppets.meet@conf.muppets.com*) and its representation by a focus user agent.

Conferencing servers mainly provide four services [3] to their related clients as shown in figure 3. The focus in a conference server acts as contact interface to interested user agents to join, modify and leave an active multi-party session. Participation can be achieved directly by sending a SIP `INVITE` request to the conference URI. Alternatively, a conference member can request the focus to invite an outstanding user agents by sending a SIP `REFER` requests. The user agent to be invited is included in a *refer-to* header field, carrying the user agent's URI. SIP messages sent by the focus are identified as those by adding an *isfocus* tag to the *contact* header field of SIP messages like shown below.

```
Contact: <sip:puppets.meet@conf.muppets.com>;isfocus
```

SIP applications that are aware of conferencing can use this indication to request notification services, e.g. user presents information, by the server. Participants that are conference-unaware just ignore this additional tag. Notifications services are supplied the *Conference Notification Server* module, and can comprise presents or conference state notification mechanisms. Notifications are sent about changes of membership or even changes of the conference state (e.g. new media parameter) [11]. The *Conference Policy Server* enforces a predefined policy, a set of conference rules (e.g. permitted participants), set up by one or more participants. The *Media Mixer* component is responsible for distributing requested media streams to each conference member, for example by using the Real-Time transport Protocol (RTP) [12].

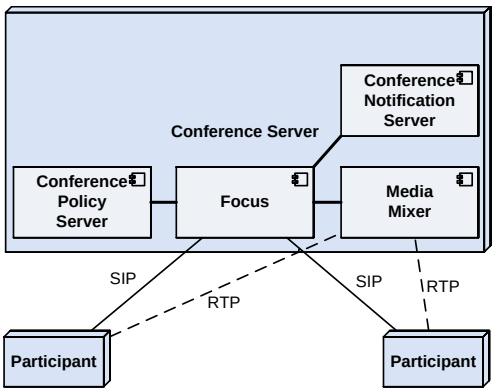

Figure 3: Service modules in a conferencing server

2.3 Distributed Conference Control

A Distributed Approach

The scalable distributed conference (SDCON) [13] protocol scheme defines node behaviors, operations and synchronization mechanisms for ad-hoc conferences in a peer-to-peer fashion. The central point of control, the *focus* of a conference, can be distributed among several participating user agents supporting the SDCON scheme. The physical split of the controlling node is done in a fully transparent way to non-SDCON able peers and appears as one entity. SDCON defines two different roles in distributed conference control scenario. First, the user agent that initially arranged and managed a multi-party conference is called the *primary focus*. Second, a participating user agent that were requested by the primary focus to become a distributed conference controller are called *secondary focus*. The only physical difference between primary and secondary focus is accounted from the uniqueness routable conference URI pointing to the primary focus. From the logical and administration, both roles provide equal behaviors. Participants can be or either have a signaling relation to a primary or secondary focus, which provide the same conferencing operations and notification services [11] and enforce the same policies. Figure 4 displays the main functionalities supported by SDCON-able user agents. The inter-focus communication is done mainly by two message flows. First, the synchronization messages shown as *focus states* and, second, the *call delegation* request messages. Call delegations happen when a focus is fully booked and needs to refer additional calls to less loaded peers. This is done by sending standard SIP compliant `REFER` requests, carrying the user agent's URI. Because calls addressed to the conference URI will be routed to the primary focus, call delegations are mainly done to a secondary focus peers. Synchronization messages will be sent on every change of state in a single focus entity, e.g., announcing the arrival of a new participant. These messages have to reach every controller to keep a consistent view to

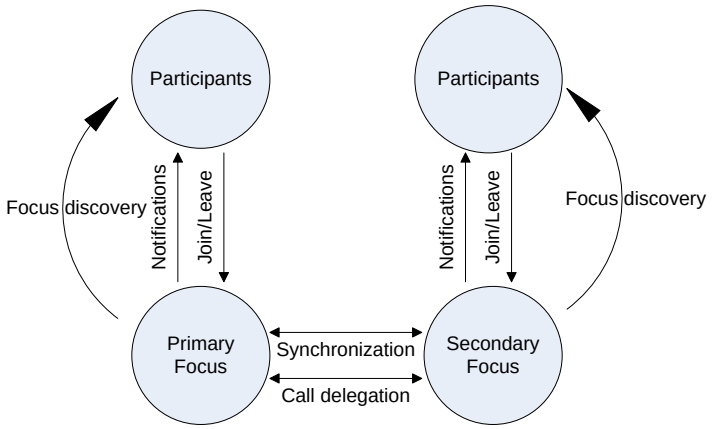

Figure 4: SDCON functionality overview

the conference. Synchronizations are sent within SIP `NOTIFY` messages, carrying an XML document describing the change event. Participants can join and leave a conference either by the primary or a secondary focus and may retrieve conference notifications by subscribing to these services. Another basic function consists in the ability to discover potential focuses with SDCON capabilities among the conference members. The *focus discovery* procedure will be executed before a focus is reaching its threshold for serving new clients. Further calls can be delegated to secondary focus peers.

ID-Locator Split at SIP Layer

To achieve a transparent distribution of the conference focus in SIP, SDCON defines a source-routing mechanism for conference participants related secondary focus peers. As the primary focus delegates a call to a secondary focus, it also transfers all already negotiated signaling and media parameter (e.g. Call-ID, SDP data). Using this information, a secondary focus peer is able to send re-invite messages to the delegated user agent, that look like as they were

originated from the primary focus. Additionally, it inserts a *Record-Route* header field, carrying the SIP URI of the secondary focus. An example SIP re-invite request is shown below:

```
INVITE sip:elmo@sesamestreet.com SIP/2.0
Call-ID: 0818@141.22.26.55
CSeq: 1 INVITE
From: <sip:puppets.meet@conf.muppets.com>;tag=134652
To: "elmo" <sip:elmo@sesamestreet.com>z
...
Contact: <sip:puppets.meet@conf.muppets.com>;isfocus
Record-Route: <sip:kermit@sesamestreet.com>
...
```

The record-route header is usually added by SIP proxies, to force further request in a SIP dialog to be routed through the proxy [2]. In this context, a conference secondary focus *kermit* adds its own SIP URI into the record-route and forces the new participant *elmo* to send further SIP requests via *sip:kermit@sesamestreet.com*. Those requests, source-routed to secondary focus peers will be intercepted and processed by them and, on change of the conference state, send notification messages to the remote focus peers. Only conference focus peers are aware of the distributed fashion of the conference control. Participants do not recognize the ID/Locator split, thus the compatibility to standard compliant SIP implementations is achieved.

3 Peer-to-Peer Overlays

3.1 Unstructured Overlays

Peer-to-Peer (P2P) overlay networks can be divided into *unstructured* and *structured* peer-to-peer systems [14]. Unstructured P2P systems like Gnutella 0.4, Freenet [15] have the characteristics, that each peer is related directly to a small number of peers participating the overlay. Resource discovery is done by flooding requests throughout the overlay network. Thus the complexity of sent messages to find a resource is greater equal $O(N^2)$. To avoid overloading the overall network by flooding resource lookup requests, these messages have a *time to live* value to reduce the the messaging load. This results in negative lookups, although the network may provides the requested resource. That makes unstructured P2P-systems not scalable for very large increasing networks and is out of scope of this paper.

3.2 Structured Overlays

Structured P2P overlay networks are using a *distributed hash table* (DHT) algorithms to provide resource location and storage. Participating nodes and the stored resources (e.g. files) are mapped into a same flat address space. This typically consists of large integer value with a range from 0 to $2^{160} - 1$ and could form a ring-like topology by performing a mathematical modulo operation on addresses space like in Chord [4]. These overlay IDs are generated by hash-function (e.g. SHA1) to obtain a unified and equally distributed identifier over the complete address space. For example a data is hashed *h('Data') = a51f2* or the IP of a participating node is hashed *h(23.119.37.77) = 1b542*. The result from the hash-function is the *key* identifier for a specific *value*. Each node in the overlay network is responsible for subset of the address space and providing an overlay routing table. They are storing the values whose keys are matching to their responsible address range. The values stored are typically the location for the requested resource, mostly the resource owner's IP address. For example, a node *n* interested in a data *d*, queries for *h(d)* to obtain the IP address of *d*'s owner *o* . *N* then queries *o* to transfer *d* directly. The very basic operations in structured P2P overlay networks are shown below:

- Join: Node participating the overlay
- Put: Add a resource into the overlay
- Get: Retrieve a resource from the overlay
- Leave: Terminate connection to the overlay

A sample structured P2P overlay network, performing `Put` operation is displayed in figure 5. The bottom of the figure displays the the underlying IPv4 topology, the top, an overlay network with flat and ring-like address space (Chord-like). Each node in the underlay has a generated it's overlay ID that is mapped at a specific position in the ring. Here, each node is responsible for the address space that is lower equal to its own ID until its the preceding node. For example, the node n_5 with the IP address *88.65.111.2* is responsible for the addresses *0Xa52e2* to *0Xa493f*. The `Put` operation proceeds as follows; The node n_0 with IP address *23.119.11.2* wants to store a data, whose hash result is *0Xa51f2*. Node n_0's routing table comprises only a subset of next hop nodes clock-wise to the address space. In Chord for example the routing table, the *finger table*, has *l* entries for a *l*-bit long identifier. On a node *n*, the *i*-th entry identifies the first node that succeeds *n* by at least the next 2^{i-1} addresses, where $1 \leq i \leq l$. Assuming that, the identifier length in the graphic is 2^{19} bits, node n_0's the 19-th routing table entry is:

$$successor(ID(n_0) + 2^{i+1}) = successor(1b542 + 2^{19-1}) = 5B542 \quad (1)$$

The result is greater than node n_2's ID and less equal than node n_3's ID, thus node n_3 becomes the next hop for the `Put` request. This routing proceeds recursive, until a node is reached, that's ID is greater equal to the key of the stored resource. In this example, node n_5 is responsible for storing the value, the IP address of data's location, of this storage request. From a view of the routing effort, the result is an average of $O(log_2(N))$ routing hops for requests in a Chord-like P2P overlay, where *N* is the number of the participating nodes [4]. This makes it highly scalable for very large increasing overlays.

3.3 Proximity Aware Overlays

Overlay network identifier are typically generated from hash-functions (e.g. SHA1 in Chord) to create unified long integer addresses for joining nodes. These IDs do normally not have any relation to the geographical position of the end system. Like indicated in figure 5, the node with IP address 23.119.37.77 is physically close to the node with IP address 88.65.111.2, but the their overlay IDs are logically 'far' away. Thus, numerical neighbors in the overlay can physically be far apart. Advanced P2P overlays [16] therefore provide mechanisms for proximity awareness. To determine a nodes relative position *p*, it measures round-trip times (RTT) against a fixed set of well known landmarks $l_0, l_1, .., l_n$. Like shown in figure 6, the node A,B and C are determining their relative positions against the landmarks L_1 to L_3. The measurement results will be ordered along the landmarks index to obtain a *landmark vector* $< l_1, l_2, .., l_3 >$. The next step is to divide the entire address space into chunks of equal-sized *regions*. The definition of a region depends on the used DHT address structure. The ring-like address space in Chord can be cut into equal slices; subtrees in Pastry can define a region or *n*-dimensional spaces in CAN. Each landmark vector permutation forms exactly one related region. A node then joins the overlay at a 'random' point in the regions its' landmark vector

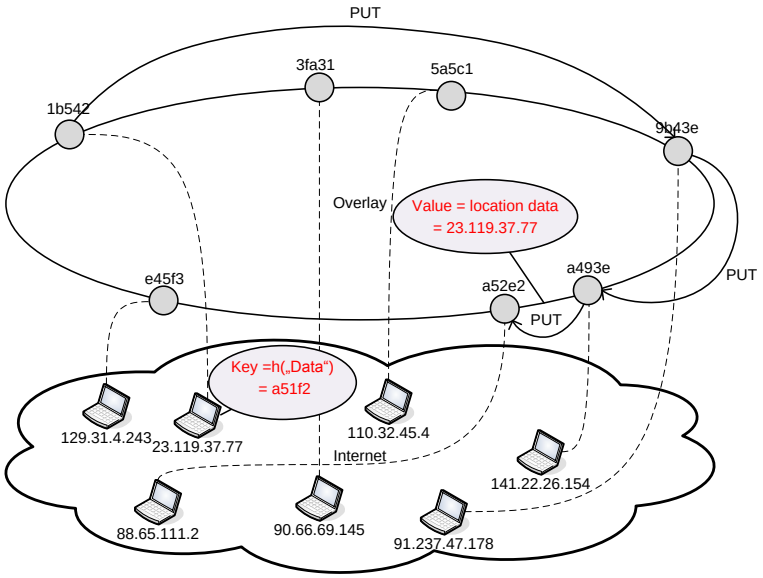

Figure 5: Structured overlay network: Put operation

belongs to. A node's relative position is then defined by its overlay ID, since every node has build its ID using the same mechanism. A disadvantage of this ID construction is that the address space is no longer unified diversified, resulting that some peers are responsible for larger address ranges than others. Overlay load-balancing algorithms can handle this problem by transferring parts of the space to less loaded peers.

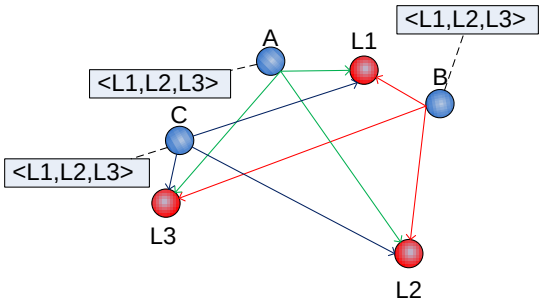

Figure 6: Estimating round-trip times against landmark-nodes

4 Approaches towards Infrastructure Independence

4.1 Virtualizing a Conference

The SDCON protocol scheme described previews in section 2.3, could be consequentially extended by using the routing and data storage functionalities of P2P networks based on *distributed hash tables*. Instead of registering the conference URI at a conference server, the owner that acts as the primary focus publishes binding mappings in the P2P service overlay. User agents attending to join the multi-party session can retrieve the contact information of the conference from the overlay to establish a SIP session. In that way it accomplishes a complete independence from server infrastructure. Means for a advanced *focus discovery* procedure, finding potential secondary focus peers can be implemented on SDCON able nodes, by advertising their capabilities in the P2P overlay. A possible way to represent such an announcement can be achieved by storing this information under a well defined *(key/value[])* strategy, dependent on the conference URI. A focus searching for a peer willing to act as a secondary focus, could retrieve the corresponding information about potential conference focuses from the overlay.

A typical problem of P2P-networks known as *churn*, is the fact that nodes are joining and leaving frequently. In case of departure of a secondary focus, the remaining focuses re-invite the now disconnected participants. As a primary focus leaves the conference or fails, an election mechanism to select a new primary focus is executed. The selected focus than has to re-register for the conference URI to be globally accessible. In a traditional conference server environment this can cause problems in to obtain the requested SIP URI, because it could be not permitted to register at the conference URI's server. To resolve this issue, a way to reassign the conference URI to the elected focus could be defined in P2P overlay.

Using a proximity-aware distributed hash table algorithm, a new participant joining a distributed conference, could decide to participate by requesting the topologically closest focus peer, like displayed in figure 7. It simply compares its overlay ID with those focus peers advertised in the overlay network, a chooses the numerical closest. This results in topologically optimized routes for the sent media streams.

4.2 Distributed Media Mixing and Routes

The SDCON approach is designed to serve as a base protocol scheme, to transparently distribute the conference management among multiple peers. The media streams could follow the signaling routed between the focus peers. Therefore, it has to be estimated which node topology can balance the computing and bandwidth effort at each focus, to provide highly scaling voice and video conferencing service. An overview of the topologies to be examined in more

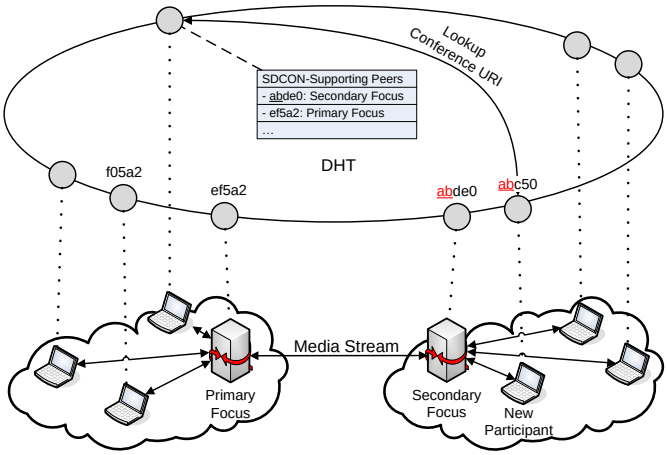

Figure 7: Discovery of a secondary focus using proximity information

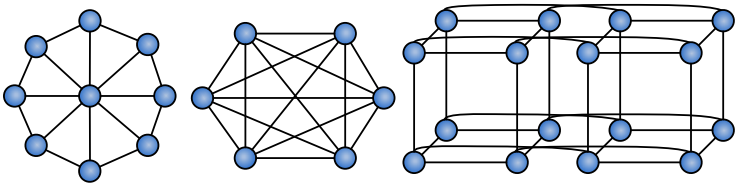

Figure 8: Examined focus topologies

detail are shown in figure 8. The examined topologies should be symmetric, of short diameter and degree to equally distribute the load, or even not, if for example some nodes are of higher computational capability. The latter approach takes the risk, that if a highly capable node fails, the supposed focus topology could crash.

5 Conclusion and Outlook

This paper presented an overview of techniques and architectures for establishing multimedia conferences with SIP, a short introduction into structured p2p overlay networks and approaches for an infrastructure-independent conferencing environment.

Signaling with SIP is based on an architecture of registrars and proxy servers, as application-layer infrastructure to facilitate end-to-end connectivity. Conferencing in a tightly coupled model, uses a central point of control, called focus in SIP. It represents the contact for a multi-party session and negotiates media parameters among the participants. It is further responsible to assure media stream connectivity. Because of this central architecture, many solutions foresee also a central, star-like media stream mixing, placed a the same physical device. A decentralized signaling scheme was presented by the SDCON approach, that transparently distributes the conference focus tasks onto multiple peers in a standard-compliant way.. This could serve as a based implementation for a distributed media mixing as well.

Structured P2P overlay networks are highly scalable for a large number of participating nodes. The underlying distributed hash tables, split the network maintenance equally onto all joining peers, each acting as router and data storage point simultaneously. Nodes and stored resources share the same flat identifier space, providing an abstract addressing scheme for to the underlying address structure (IPv4, IPv6, strings for data).

By combining the conferences with SIP and structured p2p overlays, the scaling and user location problems of conferencing with SIP can be compensated in a lean, infrastructureless manner. Following this spirit, the paper presented first approaches for a virtualized conferencing scheme, supporting the functionalities of SDCON enabled peers. This approaches seems to be interesting for further research and implementation.

References

[1] ITU-T Recommendation H.323, “Infrastructure of audio-visual services - Systems and terminal equipment for audio-visual services: Packet-based multimedia communications systems,” ITU, Tech. Rep., 2000, draft Version 4.

[2] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, “SIP: Session Initiation Protocol,” IETF, RFC 3261, Jun. 2002, updated by RFCs 3265, 3853, 4320, 4916, 5393.

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

[4] 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: ACM Press, 2001, pp. 149–160.

[5] 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*, ser. LNCS, J. Crowcroft and M. Hofmann, Eds., vol. 2233. London, UK: Springer–Verlag, 2001, pp. 14–29.

[6] 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)*, ser. LNCS, vol. 2218. Berlin Heidelberg: Springer–Verlag, Nov. 2001, pp. 329–350.

[7] M. Handley and V. Jacobson, “SDP: Session Description Protocol,” IETF, RFC 2327, Apr. 1998, obsoleted by RFC 4566, updated by RFC 3266.

[8] “The Internet Engineering Task Force (IETF),” http://www.ietf.org/, 2010.

[9] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” IETF, RFC 3550, Jul. 2003.

[10] O. Levin and R. Even, “High-Level Requirements for Tightly Coupled SIP Conferencing,” IETF, RFC 4245, Nov. 2005.

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

[12] A.-V. T. W. Group, H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” IETF, RFC 1889, Jan. 1996, obsoleted by RFC 3550.

[13] A. Knauf, T. C. Schmidt, and M. Wählisch, “Scalable, Distributed Conference Control in Heterogeneous Peer-to-Peer Scenarios with SIP,” in *Proc. of the 5th ACM/ICST International Mobile Multimedia Communications Conference (MobiMedia)*, M. Younis and C. T. Chou, Eds. Brussels, Belgium: ICST, September 2009.

[14] R. Steinmetz and K. Wehrle, Eds., *Peer-to-Peer Systems and Applications*, ser. LNCS. Berlin Heidelberg: Springer-Verlag, 2005, vol. 3485, ch. 2, pp. 12–14.

[15] ——, *Peer-to-Peer Systems and Applications*, ser. LNCS. Berlin Heidelberg: Springer-Verlag, 2005, vol. 3485, ch. 5, p. 36.

[16] S. Ratnasamy, M. Handley, R. M. Karp, and S. Shenker, “Topologically-aware overlay construction and server selection,” in *INFOCOM*, 2002.

List of Figures