

Flexible Call Control Framework Supporting Multi-party Service

A Dutta, N. H. Cheng, J. Chennikara-Varghese, S. Madhani, D. Wong, K. Young - Telcordia Technologies

Amit Patel, BAE Systems

Henning Schulzrinne, Columbia University

Abstract—In order to manage two-party and multi-party multimedia calls, a flexible call control framework is essential. The framework helps to control calls between IP end-points and non-IP devices supporting AM, FM, and GSM waveforms over an IP-based core network. Such a mechanism is more useful in a military environment where third-party control or operator-assisted calls are widely used to bridge the traditional non-IP-based end-devices with multimedia-capable IP-based terminals. This mechanism has been implemented by using a call control framework consisting of standard-based protocols such as SIP, SDP, RTP, and HTTP, entities such as B2BUA for third party call control and MCU or multicast for multi-party conferencing. This paper introduces the framework, discusses some of the functional components and explains the operational aspects of this framework.

I. INTRODUCTION

Convergence of wireless and Internet technologies provide exciting new opportunities to build flexible streaming services suitable for both military and commercial environments. One of the most desirable features, especially in a military environment with a multitude of legacy radios, is for non-IP endpoints to also be supported over a proposed IP-based core network. In an environment with both IP-based and non-IP endpoints, we therefore propose an IP-based core network that allows both the IP-based and the non-IP end-points (with various types of waveforms) to communicate with each other in a very efficient and flexible manner. However a flexible call control model is needed to manage the multi-party, multimedia calls between IP and non-IP end-points, such as PSTN or traditional analog waveform devices.

We describe here a SIP-based [2] call control model that helps set up and tear down multi-party voice over IP (VoIP) calls using several different approaches. The SIP-based multi-party framework describes the technology and methods of operation

needed to provide the multimedia capabilities across a variety of waveforms such as AM, FM, SINCGARS and JTRS. The proposed framework allows us to add/delete one or more parties to an existing call or to start a new multi-party call in a very flexible manner. Having several mechanisms available, provides us the flexibility of using this technology independent of the type of underlying network (e.g., multicast, non-multicast, PPP, Ethernet, Myrinet), and signaling protocol (e.g., SIP). Three different approaches have been designed and developed for managing the multi-party VoIP calls among dissimilar waveforms (e.g., AM/FM, GSM, SINCGARS), IP end-points and COTS IP phones, each of which uses SIP as the underlying signaling protocol. These approaches are based on MCU (Multiparty Control Unit), Multicast and RTPtranslator.

The MCU-based approach can be used as an alternative when native multicast is not available in the core of the network. As part of the MCU-based approach we have used an MCU and RTPmixer. The MCU [4] is similar to a conferencing server and each of the VoIP end-points or waveform hosts can be invited to join the MCU by a third-party controller such as SIP B2BUA (Back to Back User Agent). RTPmixer mixes the RTP stream from various sources and forwards it to each client except the source.

The Multicast-based approach provides a bandwidth-efficient solution for a bandwidth-scarce wireless medium. If the core network has multicast support or has tunnels that support multicast, this approach can be used in an efficient manner. The RTPtrans approach provides a third alternative, where SIP signaling may not be required to bridge the waveforms. B2BUA, SIP registrar and proxies can provide a scalable call control model to support communication between dissimilar end-points spanning over multiple domains. These can use any database such as MySQL for storing and managing addresses of the end-points.

In this paper we analyze the framework design, compare the three approaches, and provide the implementation details in a testbed involving Vovida's Vocal SIP user agent [5], SIP proxy, B2BUA and SIPconf-based MCU [3]. We describe the details of how AM/FM waveform hosts can convert RTP streams to analog signals and vice-versa using the VoIP gateway functionality. We also describe an application of the SIP-based call control framework to support multi-party conferencing features in a military environment.

This paper is organized as follows. Section II provides an overview of the multi-party communication models available. Section III explains third-party call control in SIP's framework. Section IV describes the functionalities of different components that are part of the architecture. Section V explains how these components interact with each other in a prototype testbed. Section VI concludes the paper.

II. MULTI-PARTY CONFERENCE MODEL

There are different types of conference models [3],[4] that can be used in practice. A conference can be defined as a "conversation space" in its abstract form. Each conversation space contains one or more participants. Participants are usually SIP user agents that send original media to or terminate and receive media from other members of the conversation space. Some of the examples involving conversation space, SIP call legs and SIP sessions are noted here. A simple two-party call is a single conversation space, a single session and a single call leg. A locally mixed three-way call has two sessions, and two call legs. A simple dial-in audio conference is a single conversation space but is represented by as many call-legs and sessions as there are human participants. A multicast conference is a single conversation space, a single session and as many call-legs as possible.

Any conferencing model consists of signaling models and media mixing models. SIP signaling can be used to set up and modify the existing calls by the end system participant. In some cases, non-participant user agents are involved in modifying the existing dialogs. Some of the examples are B2BUAs, 3pcc controllers, mixers, trans-coders, translators and relays. We have used B2BUA based

signaling model to manage multi-party conferences in this paper.

SIP also permits a variety of mixing models. In a tightly coupled conference a single SIP UA has the direct dialog relationship with each participant. In a loosely-coupled conference there is no coordinated signaling relationships among the participants. Some of the tightly coupled conference models are End System Mixing, Centralized Mixing, and Centralized Signaling but Distributed Media.

In an end system mixing model involving three users A, B and C, there are separate call legs between A, B and A, C. Here A receives the media streams from both B and C, mixes them. User A sends a stream containing A's and C's streams to B and a stream containing A's and B's streams to C. A actually handles both media and signaling mixing.

In the centralized model, all participants have a pairwise SIP signaling and media relationship with the centralized server. The media mixer part of the centralized server receives the media stream from all participants, mixes them and sends the appropriate media to other participants. The mixer usually sends the customized stream to each of the participants involved in the conference. The centralized server-based model takes the burden off the client and requires no client modification.

In a conference model with centralized signaling and distributed media, centralized server handles the signaling only but the media is still sent directly between the participants either unicast or multicast. Full mesh model is an example of a fully distributed unicast conferencing. In this model there is a signaling association between each participant and a copy of the media is sent to all the participants via unicast. This model requires a mechanism to maintain a consistent view of the distributed state across the group. Thus this model is not that popular. A multicast-based conference model involves each participant to send its media to a specific multicast address, although there is one-to-one signaling between a pair of participants within a conference. The multicast address is carried in the SDP parameters that are part of the SIP signaling between the pair of participants. Unicast receive and multicast send model assumes that the participant sends on a unicast address to the conferencing server, and then the server sends it out on a pre-

established multicast address. However, in this model each participant receives the mixed stream including its own stream.

In this paper we have tried centralized server model, multicast and RTPtrans model. We have assumed B2BUA-based third-party call control to realize these conference models.

There are several components that help set up calls between different VoIP end-points. These calls can be either two party or multi-party calls. In most cases, a call is initiated by the end-point, whereby the end-point invites another end-point using a specific signaling methodology such as SIP or H.323. The call is hung up when one of the parties decides to leave the session. During the signaling handshake, each end-point figures out the parameters of the other communicating end-point before the RTP session is established. However, there are instances where a specific two-party or multi-party call can be initiated by another agent other than the communicating parties i.e., third-party call control. This entity is called Back-to-Back UA. Back-to-Back UA communicates with each participant and acts like a signaling mixer to pass on the end-point parameters to the end-points.

III. THIRD-PARTY CALL CONTROL

Third-party call control allows one entity called the controller to set up and manage a communication relationship between two or more other parties. Third-party call control (referred to as 3pcc) [1] is often used for operator services. Many SIP-based services are possible through third-party call control. These include the traditional ones on the PSTN, but also new types of service such as click-to-dial. Click-to-dial allows creating a call between two users. The calls can be between two phones, a phone and an IP host or two IP hosts. A controller is a Back-to-back SIP User Agent that wishes to create a session between two user agents. In a military environment, a third-party controller can be instructed by a control and reporting segment called CRS. There are many call flows possible to implement a third-party call control. Reference [1] lists some of the call flows that allow third-party call control mechanism. Figure 1 shows an example of SIP-based B2BUA assisted call flow between two end-user agents. B2BUA helps to establish a multimedia session between two end-points by exchanging the SDP parameters such as IP address,

port parameters, CODEC parameters etc. Initially B2BUA sends a separate INVITE signal to each user agent with null SDP or no SDP. User Agent sends the OK signal back with the associated parameters of the end point in it. These parameters are passed on to the respective end point as part of the ACK signal. Because of the timing associated with both the INVITEs, the first user agent (UA1) does not get the parameters in its first ACK. Thus B2BUA has to send the second Re-Invite after it has obtained the parameters from the OK sent by UA2. Second Re-Invite contains the desired parameters of the end points so as to complete the handshaking between the user agents. RTP session is established after the signaling is complete.

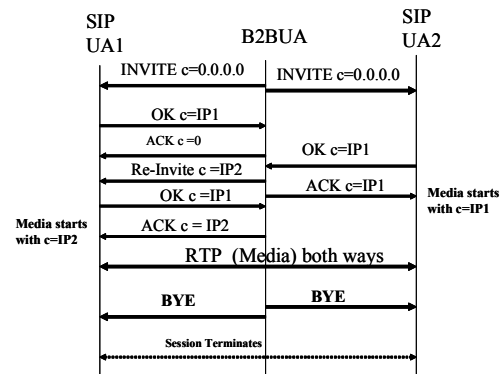


Figure 1: B2BUA-based call flow

Depending upon how many types of media (audio, video, and data) are selected, that many media are invoked. The session is terminated as the B2BUA sends BYE to each communicating party. Here one or both of the end points can also be a conference server.

IV. FRAMEWORK COMPONENTS

This section lists some of the components of the flexible framework described in the paper.

Waveform Host: Waveform Host (WH) is built upon SBC Quad Jumpgate 2814 and runs on Yellow Dog Linux operating system (YDL). There can be multiple kinds of waveform hosts depending upon the type of waveform it is supporting. The waveform host has IP interface and can be of types AM, FM, GSM, and SINCGARS. All these connect to the IP core network via its IP interface. Each WH has a non-IP side for connection to different types of analog waveforms. Each waveform host converts the AM/FM waveforms to desired IP stream by using its

inbuilt A/D converter. An example of AM/FM waveform host is shown in Figure 2. This interface gives SIPUA component control of VoIP through VoIP parameters. VoIP component of the waveform host takes digitized audio, encodes it with a codec into frames, and then packetizes the audio frames into RTP packets to be sent over IP, and vice versa for receiving RTP packets. The VoIP component also provides an interface to a SIPUA component if call signaling is required or if compatibility with SIP is needed in a JTRS (Joint Tactical Radio System) application.

Figure 2 shows a data flow diagram showing how the SIP UA components of the AM, FM waveform hosts get connected to an MCU.

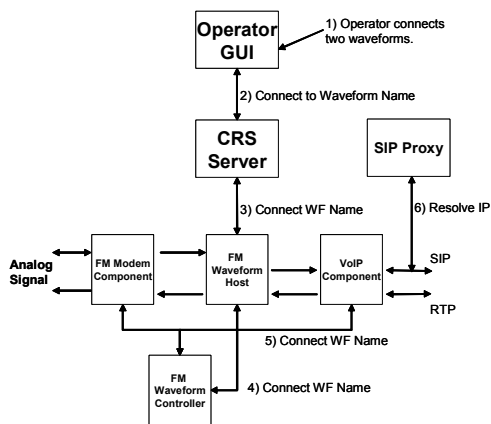


Figure 2: Waveform Host Components

SIP User Agent: SIP user agent runs on any end point IP device that wishes to communicate with each other. These end-devices can be either IP-based host or AM-, FM-based waveform host having IP access to the core network. Each SIP user agent has the ability to support multimedia calls, but it depends upon the multimedia capability of the hardware. For example a multimedia terminal can support audio, video and data, whereas AM/FM waveform hosts can support voice only. SIPUA component will be able to send INVITE to other SIPUAs, and receive INVITE from SIPUAs and B2BUA. The SIPUA component of the B2BUA, will also register with the SIP Proxy on instantiation. The SIPUA sends VoIP parameters in SDP format when performing an INVITE. Once a call is made, and settings have been negotiated the SIPUA will use a JTRS port interface to set the VoIP parameters.

SIP Proxy: SIP proxy is another signaling component which acts like an application-level router. It can route a SIP-based call either originating from an end point or from a back-to-back user agent. The end point does not need to have the knowledge of the IP address associated with the other end point to set up the call. The end point uses a SIP URI (Unique Resource Identifier)(e.g., sip:bob@xyz.edu) that can be used by the SIP proxy to route the call properly. There can be a SIP proxy dedicated to a specific domain. If a user does not belong to a specific domain, the respective SIP proxy can forward this call to the SIP proxy belonging to the correct domain. Once the call gets to the proxy in the designated domain, it gets routed to the end point in that domain. Direct calling is another alternative to proxy-based calling. The B2BUA will be tasked by the CRS to initiate calls between SIPUA components. It will interface with SIPUA components through a common SIP port in order to send/receive SIP/SDP packets.

SIP registrar: SIP registrar is a part of SIP server. It keeps an association of user's URI and the contact address such as an IP address. A SIP user agent on a specific waveform host or multimedia terminal can register its IP address with the SIP registrar. SIP registrar works in conjunction with the SIP proxy to route the call to the specific destination based upon its contact address. SIP registrar can interact with other types of databases such as whois server, finger server and LDAP server. SIP registrar can also register several instances of conference server.

MCU: MCU is defined to be Multi-Point Control Unit. It is a conferencing server that consists of a signaling and media mixing module. An MCU can be implemented using both SIP and H.323 based signaling protocol.

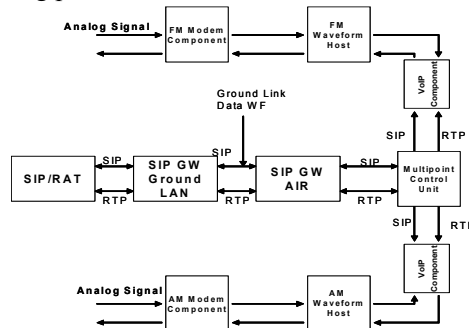


Figure 3: MCU based call with dissimilar Waveform

In H.323 architecture MCU consists of a multi-point controller to handle signaling and control exchanges with every participant in the conference, and a multi-point processor handles the mixing and filtering of different media streams. In a SIP-centric environment, the signaling module provides multiple instances of a SIPUA and the media mixing module receives and sends RTP media streams from and to participants. There is a signaling association between SIPUA on each waveform host and the SIPUA associated with the MCU. RTP streams from each of the waveform host get forwarded to the rest of the participants. Figure 3 shows how MCU can help set up a call between two dissimilar waveforms and an IP end-point. An example of how MCU takes part in the multi-party call has been illustrated in Section V.

RTP Mixer: An RTP mixer is an intermediate system that receives RTP packets from a number of sources and combines them into a single output possibly changing the coding before forwarding it to the rest of the participants.

Back-to-Back UA (B2BUA): B2BUA is a SIP entity that acts like a third party call controller to set up and delete calls between the end-points. It receives the end-point parameters and possibly copies these SDP parameters in the signaling message to be sent to the other communicating end-point. A B2BUA can also help providing session mobility.

CRS (Control and Reporting Segment): Control and Reporting Segment is an entity that interfaces with the SIP registrar to find out the available end-points and conference servers within a specific domain. Once it has the knowledge of available end-points it can send the HTTP control command to set up/delete the multimedia calls between the end-points or between the end points and MCU.

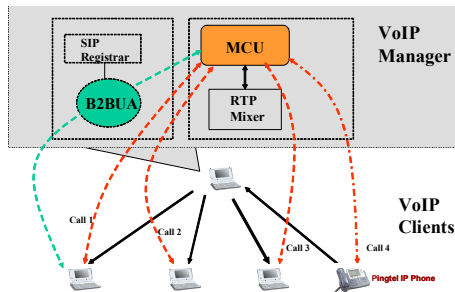


Figure 4 – Multicast VOIP Infrastructure

Figure 4 shows a VoIP infrastructure where CRS instructs the B2BUA to set up a multicast conference between the registered participants. Here the SIP signaling is unicast, but the SDP parameters have multicast address, thus upon completion of the SIP signaling date is sent over multicast address.

Figure 5 shows an example of MCU-based multiparty call setup and interaction between different components.

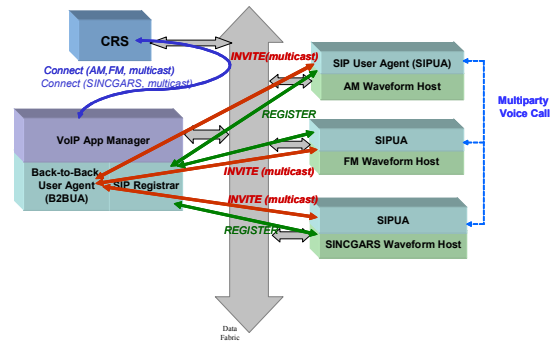


Figure 5: MCU based call interaction

V. TESTBED PROTOTYPE

We have prototyped several components under Yellow Dog Linux environment using Apple Power books and 2841 chassis. 2841 has a Myrinet backplane and has multiple boards plugged into it. The first board acts like a master and the rest of the boards boot off the master board.

Basic Features: The following lists some of the basic operational features associated with the third-party call control for a single domain in the military environment. It provides the ability to figure out the list of conferences and the users that are part of a specific conference by obtaining the MCU status file. It makes use of the mcu-registry file and sip-registry file to setup ad delete users from the conference.

In a multiple domain scenario CRS communicates with DNS to determine the available proxies in each domain and queries these proxies to find out the addresses registered with each proxy. It can use register function if it is UA-enabled, forms a set of IDs to populate its MySQL database, and associates

the ids with respect to the proxy. Here are some of the assumptions for multi-B2BUA environment

- Each B2BUA may synchronizes its databases with the adjacent B2BUA
- One B2BUA may not have the ability to set up a call to the other end-point under a different domain
- In that case the B2BUA can go through the proxy or other B2BUA
- Each B2BUA knows about the existence of other B2BUA via some discovery mechanism
- If the end-point address (2136661111) is not meant for that specific B2BUA (213555), it will forward the request to the correct B2BUA (213666) who in turn will forward it to the right end point that is registered with that specific B2BUA.

MCU Registry Operation: CRS gets the MCU registry file in ASCII format from each domain via HTTP request. MCU registry provides the conference names and associates URIs for addressing purposes. Each conference can be provisioned with multiple URIs (ports) to provide multiple instances of SIP UA. Each end-point has to use different URI while connecting to the same conference.

Several instances of conference server are illustrated below. Each SIPUA on the waveform host connects to the conference server using a specific instance of the conference server. It is also possible to have multiple conferences to be part of the same MCU such as Conf1 and Conf2. Different groups of users can be part of either conference. The following shows the example of two different conferences and their associated URIs.

```
Conf1@pl1:s1=2135551000
Conf1@pl1:s2=2135551001
Conf1@pl1:s3=2135551002
Conf1@pl1:s4=2135551003
```

```
Conf2@pl1:s1=2135552000
Conf2@pl1:s2=2135552001
Conf2@pl1:s3=2135552002
Conf2@pl1:s4=2135552003
```

SIP Registry: SIP registry provides the list of all the participants that are present within a domain. CRS determines the IP addresses of the active payloads via DNS, CRS gets the SIP registry ASCII file from the respective payload via http. Registration method provides the URI of the participants (e.g.,

2135551111, 2135552222, 2136661111 for the waveform hosts such as w1@PL1, w2@PL1, w1@PL2 and the associated IP addresses of the respective gateways). XML maps the URIs to the names such as AM, FM, GSM for local use.

MCU-status: CRS pulls an ASCII file from the payload via HTTP . As a participant joins/leaves the conference current participant list gets updated. The following shows an example

A file with empty conference may look like this:

```
File1:      Conf1:
           Conf2:
```

As UA1, UA2 join conf1, UA3, UA4 join conf2, the file may look like as follows

```
File1: Conf1:<IP address 1,UA
identifier1>:<IPaddress2,UA identifier2>
Conf2:<IPaddress3, UA
identifier3>:<IPaddress4,UA identifier4>
```

As UA2 leaves conf1, UA4 leaves conf2 status file may look like this

```
File1: Conf1:<IP address1,UA Identifier1>
Conf2:<IP address 3, UA identifier3>
```

Single Node Operation: Following are some of the examples of how CRS can instruct a single B2BUA so that it can add and delete waveform hosts from two conferences conf1 and conf2. W1 has a URI 2135551111 and conference conf1 can have multiple instances as shown before.

Adding waveform host W1 to conf1 can be realized as follows.

```
http://payload:8001/3pcc.html?p0=<W1>&p1=<conf1:s1>
```

Deleting a waveform host from conf1 can be realized as follows.

```
http://payload:8001/3pcc.html?p0=<W1>&p1=<conf1:s1>&cmd=HangUp
```

Similarly another waveform host W2 (2135552222) can be connected to or deleted from the same conference (conf1) in the following manner respectively

```
1)http://payload:8001/3pcc.html?p0=<W2>&p1=<conf1:s2>
```

```
2)http://payload:8001/3pcc.html?p0=<W2>&p1=<conf1:s2>&cmd=HangUp
```

Communication with multiple B2BUAs: In this case CRS will help establish the communication between the end-points belonging to each B2BUA. Each SIP server within a domain knows the existence of the other domains either via DNS or via pre-provisioning. Each B2BUA is associated with a numeric prefix such as 213555, or 213666 so that it can differentiate one B2BUA from another. CRS gives control command to one of the B2BUAs that is in the line of sight. B2BUA looks up in its database to find out if the end-point IP address belongs to its own domain or it is part of the second B2BUA. It sends the call to the other B2BUA via Inter-nodal communication link where the other end-point has registered. Waveform W1@PL1 has a URI such as 2135551111 and conference instance at the second domain (conf1@PL2:s1) is identified as 2136661000. Adding and deleting W1 waveform of domain1 to and from a conference at the second domain can be realized as follows.

http://payload:8001/3pcc.html?p0=<W1@PL1>&p1=<conf1@PL2:s1>

http://payload:8001/3pcc.html?p0=<W1@PL1>&p1=<conf1@PL2:s1>&cmd=HangUp

Figures 6 and 7 show the interaction of several components and command sequence when multiple B2BUAs are involved to set up a call between entities over multiple domains. SIP proxies in the respective domains may also inter-operate with B2BUA while providing multi-party conferencing. However in this scenario respective SIP proxies should be able to discover each other's IP address through discovery mechanism such as DNS "srv" or protocols such as SLP (Service Location Protocol).

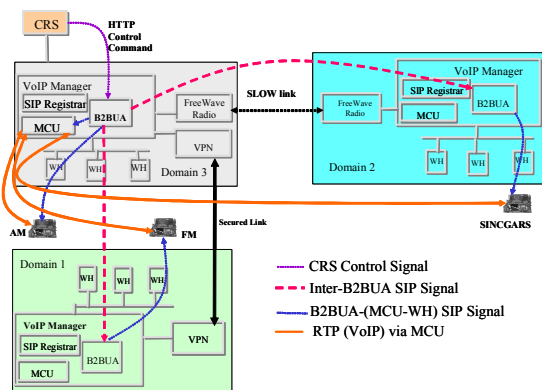


Figure 6: Call control using multiple B2BUAs

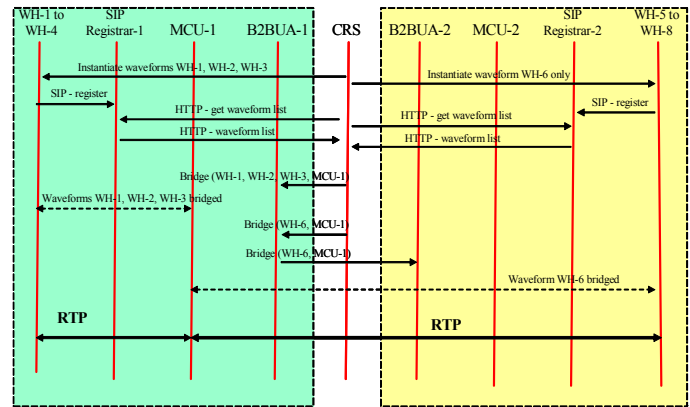


Figure 7: Flow sequence for a conference bridge

VI. CONCLUSIONS

We presented a SIP-based call control framework that can help manage two-party and multi-party multimedia calls using an IP-based infrastructure. The end-points can be traditional communication devices supporting AM, FM or GSM waveforms. These can communicate with IP-based multimedia terminal and can be controlled via its IP interface using unique URI scheme. It can also help manage different types of conferencing models such as MCU-based and multicast-based spanning over multiple administrative domains. Such a framework supporting third-party call control and various types of conferencing models will be beneficial both in military and commercial environment.

REFERENCES

- [1] J. Rosenberg et al, "Best Current Practices for Third Party Call Control in the Session Initiation Protocol," draft-ietf-sipping-3pcc-06, Work in Progress, IETF
- [2] J. Rosenberg et al, Session Initiation Protocol, Request for Comments, 3261, IETF
- [3] Kundan Singh, Gautam Nair, Henning Schulzrinne, "Centralized Conferencing Using SIP", Proceedings of IPTel 2001
- [4] Roham Mahy et al., "A Call Control and Multi-party usage framework for the Session Initiation Protocol (SIP), IETF Work in progress
- [5] www.vovida.org