Centralized Conferencing in the IP Multimedia Subsystem: from theory to practice

A. Amirante, A. Buono, T. Castaldi, L. Miniero and S. P. Romano

Original scientific paper

***Abstract*—In this paper we present a conferencing architecture compliant with the IP Multimedia Subsystem (IMS) specification. To the purpose, we embrace a practical approach, by describing an actual implementation of an open source centralized video-conferencing system, called CONFIANCE, capable to offer advanced communication experience to end-users through the effective exploitation of mechanisms like session management and floor control. CONFIANCE has been designed to be fully compliant with the latest standard proposals coming from both the IETF and the 3GPP and can be considered as an outstanding example of a real-time application built on top of the grounds paved by the SIP protocol. We will discuss in the paper both the design of the overall conferencing framework and the most important issues we had to face during the implementation phase.**

***Index Terms*—IP Multimedia Subsystem, Video Conferencing, Floor Control**

I. Introduction

Conferencing can nowadays be considered by providers as an extremely challenging service, since it imposes a number of stringent requirements to the underlying network infrastructure. First, the intrinsic multimedia nature of a conference (which typically involves a combination of audio, video, instant messaging, desktop sharing, etc.) requires coping with complex issues like session management and floor control. Second, the real-time features of conference-based communication call for an appropriate level of *Quality of Service* (QoS).

In order to effectively support advanced services like conferencing, the third generation Partnership Project (3GPP) is currently standardizing the so-called IP Multimedia Subsystem (IMS), an architecture aimed at providing a common service delivery mechanism capable to significantly reduce the development cycle associated with service creation across both wireline and wireless networks. The envisaged portfolio of IMS services includes, besides the already mentioned conferencing service, other advanced IP-based applications like Voice over IP (VoIP), online gaming and content sharing.

The main challenge for the IMS is that all such services are to be provided on a single, access-agnostic, integrated infrastructure capable to offer seamless switching functionality between different services. This is achieved through the adoption of the principle of separation of concerns, which reflects in the definition of a number of core IMS components (such as Call/Session Control Function – CSCF, Home Subscriber Server – HSS, Media Resource Function – MRF and Application Server – AS), each in charge of looking after a specific function of the overall system. The identified components must also be scalable and able to provide advanced features, like *five nine* reliability.

The goal of this paper is to present an actual implementation of *CONFIANCE*, an IMS-compliant conferencing architecture that has been conceived at the outset as a playground useful both for protocol testing and for field trials and validation. We will first present our architecture from a high-level perspective, in order to highlight the mapping between IMS logical functions and the actual system components. Then, a more in-depth view of such components will be provided, so to allow for a better explanation of the most challenging choices we had to make during the implementation phase.

The paper is structured as follows. Section II helps position our work by providing useful information about the reference context, as well as about the motivations behind our contribution. An IMS-compliant architecture for moderated video conferences is depicted in section III. Implementation details are illustrated in section IV, whereas in section V we deal with related work. Finally, section VI provides some concluding remarks, together with information about our future work.

II. Context and Motivation

Nowadays the most widespread signaling protocol for IP networks is the Session Initiation Protocol (SIP) [1]. It provides users with the capability to initiate, manage, and terminate communication sessions. SIP natively allows multi party calls among multiple parties. However, conferencing does represent a more sophisticated service that can be seen as an extension of multi party calls where audio is just one of the possible media involved. For example, the conferencing service may provide video functionality as well as instant messaging, files and presentation sharing or even gaming.

Furthermore, the conferencing service provides the means for a user to create, manage, terminate, join and leave conferences. Finally, it provides the network with the ability to deliver information about these conferences to the involved parties.

Over the last few years, standardization efforts have been devoted to conferencing related matters by international bodies like the IETF, the 3GPP and OMA.

The Internet Engineering Task Force (IETF) is an open international community concerned with the evolution of the Internet architecture and protocols. Within the IETF the Centralized Conferencing (XCON) working group is explicitly

Manuscript received December, 2007 and revised February, 2008.

This Paper was presented as part at the Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN) 2007

Alessandro Amirante, Tobia Castaldi, Lorenzo Miniero and Simon Pietro Romano are with the University of Napoli Federico II, Via Claudio 21, 80125 Napoli, Italy. Email: {alessandro.amirante, tobia.castaldi, lorenzo.miniero, spromano}@unina.it

Alfonso Buono is with the CRIAI Consortium, P.le E. Fermi 1, 80055 Portici (NA), Italy. Email: {alfonso.buono@criai.it}

1845-6421/08/7182

focusing on multimedia conferencing. Furthermore, there are a couple of working groups whose standardization activity also deals with conferencing related issues, namely Session Initiation Proposal Investigation (SIPPING) and Media Server Control (MEDIACTRL).

The very first framework for multi-party conferencing based on the SIP protocol was developed by the SIPPING WG [2].

This framework defines a general architectural model, presents terminology, and explains how SIP is involved in a tightly coupled conference. Taking inspiration from the work carried out in SIPPING and willing to release any constraint about the signaling protocol, the recently born XCON WG is working hard on the definition of both a reference framework [3] and a data model [4] for tightly coupled conference scenarios.

The envisaged architecture is based upon a centralized management component, called *focus*, conceived as a logical entity which integrates both signaling and control features. Specifically, it acts as an endpoint for each of the supported signaling protocols and maintains a call signaling interface between each participant client and the so-called *conference object* representing a conference at a certain stage (e. g. description upon conference creation, reservation, activation, etc.). Thus, the focus is responsible for all primary conference membership operations. At present, XCON has specified the moderation protocol, the so-called Binary Floor Control Protocol (BFCP) [5]. BFCP enables applications to provide users with coordinated (shared or exclusive) access to resources like the right to send media over a particular media stream. Recently, the *MEDIACTRL* working group has started working on the definition of a framework for the remote control of a Media Server (MS) from an Application Server (AS). The AS acts as a decision point, since it hosts the business logic. The MS, on the other hand, behaves like an enforcement point, which the AS controls through messages sent across an ad hoc defined Control Channel. The messages exchanged between AS and MS belong to specific Control Packages (e.g. conferencing, basic IVR – Interactive Voice Response, etc.).

The 3^{rd} Generation Partnership Project (3GPP) actually represents a collaboration agreement among a number of regional standard bodies, born with the main objective of developing Technical Specifications for a third-generation mobile system based on GSM. Recently, the 3GPP has worked on the specification of a tightly-coupled conferencing service. Both the requirements and the architecture for such a service have been defined [6]. The cited document indeed represents a sort of integrated specification within the IMS, aimed at harmonizing the combined use of existing standard protocols, like the Session Initiation Protocol (SIP), SIP Events, the Session Description Protocol (SDP) and the Binary Floor Control Protocol (BFCP).

Coming to the Open Mobile Alliance (OMA), we just point out that it represents the leading industry forum for developing so-called *mobile service enablers* on the extensive 3GPP IMS architecture. The key to the success of such service enablers resides in the market-driven approach to deployment, as well as in interoperability. The OMA Conferencing solution is a primary example of an application built on top of the service enablers. To date, OMA has standardized conference models for both Instant Messaging [7] and Push to Talk [8].

A. The IMS architecture

Fig. 1 shows the architecture for the 3GPP IMS conferencing service. Both IMS entities and IMS interfaces are showed in the picture. In the following of this subsection we briefly expand on the components which are relevant to our project.

The *User Equipment* (UE) implements the role of a conference participant and may support also the floor participant or floor chair role (the difference between such roles will be clarified in section IV).

The UE might be located either in the Visited or in the Home Network (HN). In any case, it can find the P-CSCF via the CSCF discovery procedure. Once done with the discovery phase, the UE sends SIP requests to the *Proxy-Call Session Control Function* (P-CSCF). The P-CSCF in turn forwards such messages to the *Serving-CSCF* (S-CSCF). In order to properly handle any UE request, the S-CSCF needs both registration and session control procedures (so to use both subscriber and service data stored in the *Home Subscriber Server* – HSS). It also uses SIP to communicate with the *Application Servers* (AS). An AS is a SIP entity hosting and executing services (in our scenario, the AS clearly hosts the conferencing service).

The *IP Multimedia Service Control* (ISC) interface sends and receives SIP messages between the S-CSCF and the AS. The two main procedures of the ISC are: (i) routing the initial SIP request to the AS; (ii) initiating a SIP request from the AS on behalf of a user. For the initiating request the SIP AS and the OSA SCS (*Open Service Access - Service Capability Server*) need either to access user's data or to know a S-CSCF to rely upon for such task. As we already mentioned, such information is stored in the HSS, so the AS and the OSA SCS can communicate with it via the *Sh* interface.

In a SIP based conferencing scenario, the MRFC (*Media Resource Function Control*) shall regard the MRFP (*Media Resource Function Processing*) as a mixer. In fact, the MRFP hosts both the mixer and the floor control server. When the MRFC needs to control media streams (creating a conference, handling or manipulating a floor, etc.) it uses the *Mp* interface. This interface is fully compliant with the H.248 protocol standard. The MRFC is needed to support bearer related services, such as conferencing. The focus, conference policy server and media policy server are co-located in an AS/MRFC component in the 3GPP framework. S-CSCF communicates with MRFC via *Mr*, a SIP based interface.

In this scenario the AS/MRFC shall implement the role of a conference focus and a conference notification service. MRFC may support the floor control server role, the floor chair role or the floor participant role.

III. Designing an IMS compliant video conferencing architecture

We started our design from an architectural perspective of the service we wanted to achieve, that is an advanced

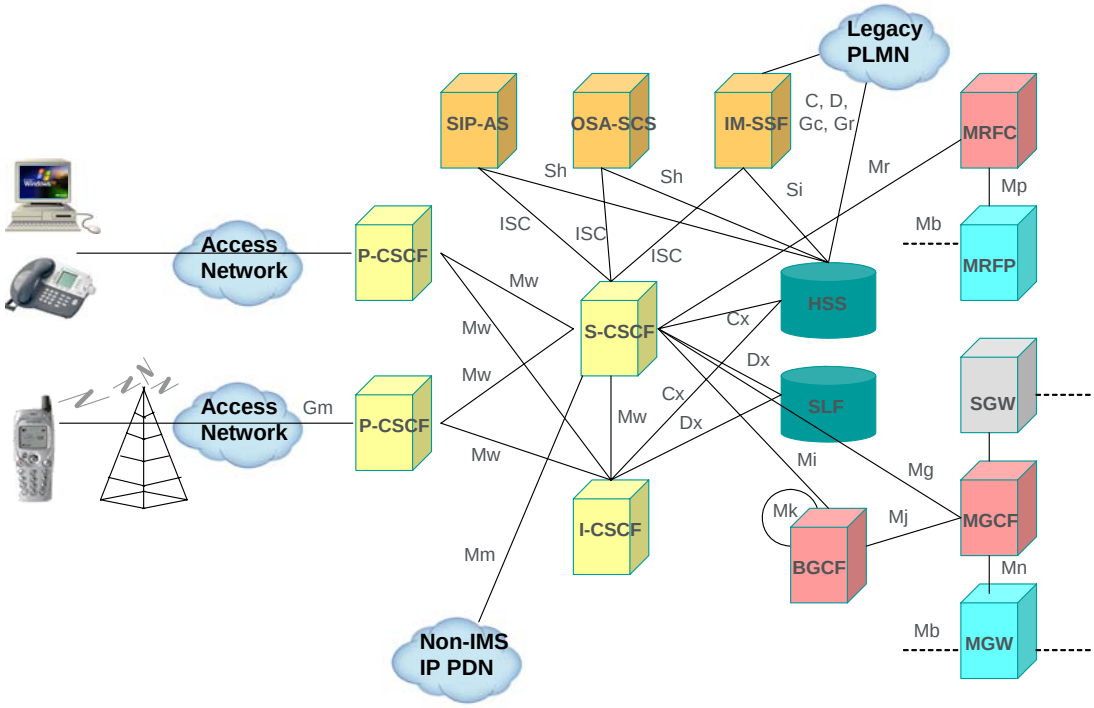

Fig. 1. The IMS Architecture

conferencing application. The first step was obviously identifying and locating all the IMS logical elements which would be involved in the above mentioned scenario. We then investigated the possibility of replicating, or at least replacing, such elements with existing real-world components. All the presented steps are described in detail in the following sections.

A. Identifying the required IMS elements

To identify the required elements, a bird's eye view of the whole IMS architecture, as shown in in Fig. 1, is first needed. Then, considering the desired conferencing scenario, identifying such logical elements can be easily accomplished.

First of all, the scenario clearly addresses two distinct roles, a server side (the elements providing the service) and a client side (all users accessing the service). We referred to these roles in identifying the elements.

Starting from the client side, the very first mandatory element that comes into play is of course the User Equipment (UE), which has to be both SIP compliant and XCON enabled in order to correctly support conferencing.

At this point, the other involved elements can be found by following the flow of messages originated by users. This indicates we need to have the P-CSCF, which may behave like a proxy, accepting incoming requests and forwarding them to the S-CSCF. Hence, S-CSCF and HSS are the next selected elements, which are to take care of many important tasks, the most relevant ones being checking users access and authorization rights, handling session control for the registered endpoint sessions and interacting with Services Platforms for the support of services.

Of course, so far the selected elements only deal with the signaling plane of the conferencing scenario. Considering that we need elements performing the service itself, the focus must then be moved to floor management, media streaming and control. This leads us in selecting additional elements to the design. To handle the service, a SIP-AS (SIP Application Server) as defined in [9] is to be used. This AS will be in charge of managing conferences (e.g. creating, modifying, deleting them), and in general of all the business logic, which includes policies, related to the scenario. These policies include Floor Control, which implies that the AS will have to also manage access rights to shared resources in our conferencing framework. The media streams are manipulated and provided, according to the IMS specification, by a couple of tightly coupled elements called MRFC (the controller) and MRFP (the processor). The MRFC controls the media stream resources in the MRFP, interpreting input coming from the AS, then instructing the MRFP accordingly. The MRFP is the element which actually provides the resources, by offering functionality like mixing of incoming media streams (in our case, audio and video streams) and media stream processing (e.g. audio transcoding, media analysis). Considering the XCON framework is conceived to be agnostic with respect to the signaling protocol used to access the service, a specific element is needed as a gateway towards other technologies. This can be accomplished by the MGCF, which performs the interworking with the PSTN, while controlling the MG for the required media conversions. As a final step, the MGW helps perform the interworking with the PSTN, at the same time controlling and reserving the resources required by the media streams.

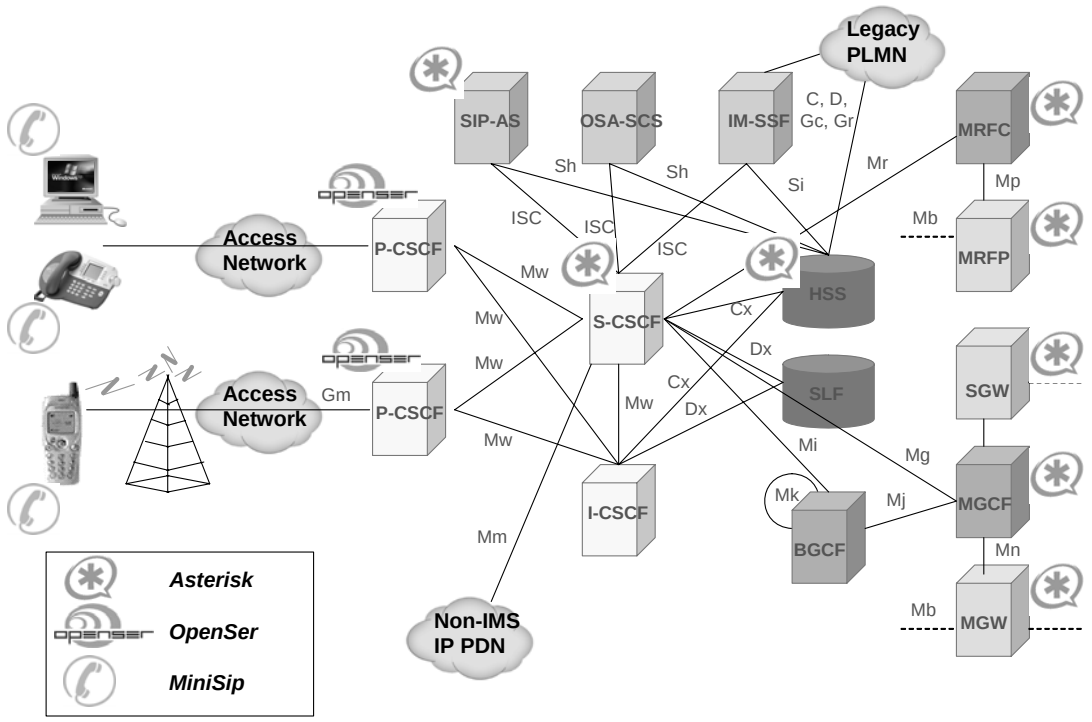

Fig. 2. IMS Elements Mapping

B. Elements mapping

While the first step was to identify the elements required by the service, our next step was to look for possible real-world components capable to realize the needed functionality. In our architecture, we found many of these components in the open source community. However, in many cases we had to implement our own components, either from scratch or based on existing open source projects. Fig. 2 shows a graphical view of the mapping we made between the addressed IMS elements and the chosen components. As to the UE, we opportunely modified an open source SIP client called *Minisip* (`http://www.minisip.org/`), in order to make it capable to handle BFCP protocol messages. The P-CSCF was replaced by a fully compliant SIP Proxy server called *OpenSER* (`http://www.openser.org/`). The S-CSCF and HSS elements have been realized by exploiting the functionality provided by the well known open source PBX called *Asterisk* (`http://www.asterisk.org`). Asterisk actually was able to provide us with many of the required IMS functions. In fact, it already included a module, called *MeetMe*, capable of providing very basic conferencing functionality. Its modular approach for what concerns applications allowed us to enhance this component with all the functionality needed to realize the role of the SIP-AS as played in our architecture, thus making it capable to manage conferences. Furthermore, the roles of the MRFC and MRFP components are partly played by another pair of ad-hoc modified Asterisk modules capable to provide media management, streaming and floor control. Additional MRF-related functionality, especially for what concerns video, have been realized by implementing from scratch a remotely controllable VideoMixer. Finally we replaced the MGCF and MGW components with a native Asterisk component performing the interworking with the PSTN, including all the related activities.

Starting from the considerations above, in the next section we will describe in detail the implementation choices behind our architecture.

IV. CONFIANCE: AN OPEN SOURCE IMPLEMENTATION OF THE CONFERENCING ARCHITECTURE

This section is focused on presenting our actual implementation of an open platform for the support of the already introduced IP-based conferencing scenario. We first realized this framework, which we called *CONFIANCE (CONFerencing IMS-enabled Architecture for Next-generation Communication Experience)*, in the framework of a collaboration activity involving the University of Napoli and Ericsson's Nomadic Lab in Helsinki. The aim was to try and take into account the most recent proposals under development inside the several interested standardization communities. This led us, starting from the IMS-compliant design described in the previous sections, to implement an XCON-compliant video conferencing service, in order to provide advanced capabilities, like conference management and moderated access to the available resources. Great inspiration for this task came to us from the work ongoing inside the IETF, especially in the XCON and MEDIACTRL working groups.

As already introduced in section II, the XCON framework (see Fig. 3) defines a suite of conferencing protocols, which are meant as complementary to the call signaling protocols, for building advanced conferencing applications and achieving complex scenarios. These protocols aim at providing means to

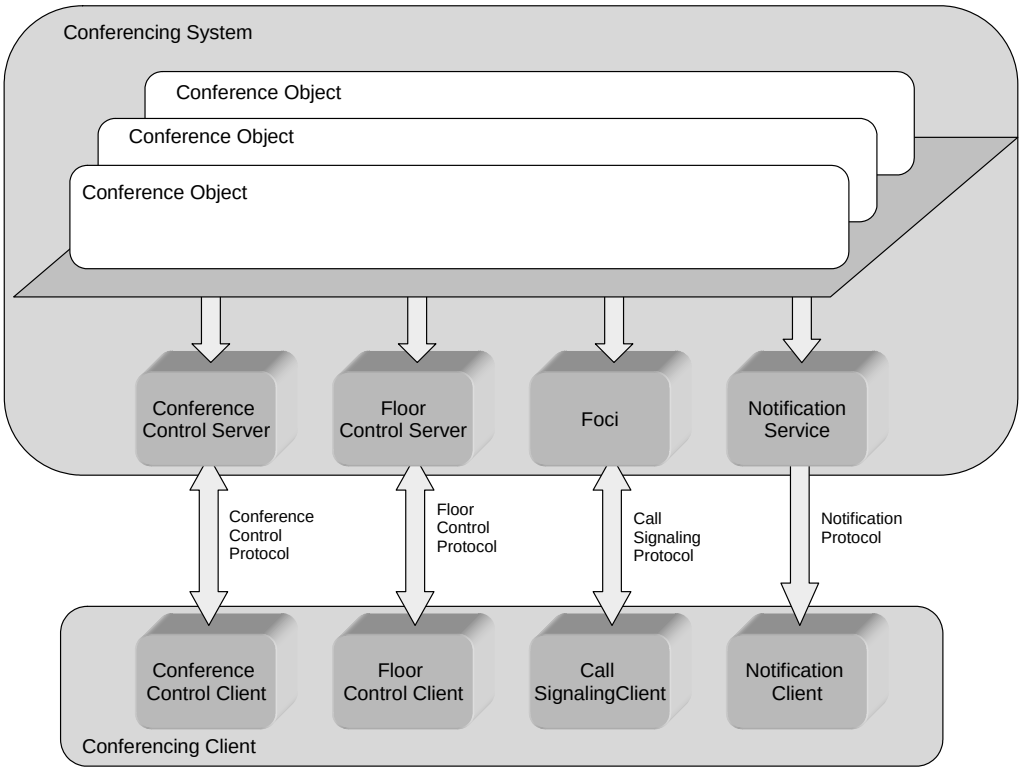

Fig. 3. Logical decomposition of the XCON Conferencing Framework

manage conferences in all their facets. Among them, the so-called BFCP (Binary Floor Control Protocol) deserves special attention. BFCP, in fact, enables conferencing applications to provide users with coordinated (shared or exclusive) access to the resources which have been made available. This *coordinated access* stands for the capability to apply and enforce policies upon media delivery, by opportunely managing the access to a set of shared resources, such as the right to send media over a particular media stream. A typical use case is a participant willing to talk in a lecture-mode conference, who has to submit a request to a designated chair in charge of taking a decision accordingly.

According to the protocol specification, each shared resource or set of resources can be associated with a logical entity called a *floor*. This floor is defined as a permission to temporarily access or manipulate the associated set of resources. A floor becomes the token that different actors in the specification refer to in their interaction. One of these actors is a logical entity called *chair*, which is made responsible for one or more floors. Its main task is managing incoming requests for the floors it is assigned to, by accepting, denying or revoking them. The requests come from clients of a conference, who can make floor requests on a transaction-by-transaction basis to the Floor Control Server, thus asking for the permission to access a specific set of resources. The server handles a set of queues according to the available floors and the associated policies, and if needed forwards incoming requests to the designated chair, asking her/him for a decision about them. Chairs are also offered more complex functionality, e.g. to actively revoke a floor from a participant who may be abusing it. It is worth noting that, even though BFCP offers a way to coordinate access to resources, how these resources are associated with floors and the policies a Floor Control Server may follow, as well as the queue scheduling it may enforce, are outside the scope of its specification.

The realization of such an XCON-compliant architecture led us to work both on the client and on the server side, with special focus on all the communication protocols between them and their scenarios of interaction. The client side work included the implementation of both roles envisaged in the architecture, namely the simple participant and the chair. On the server side, we implemented the roles of the focus, as defined in [3], of the Floor Control Server, and of the Media Server. To make the client and server sides interact with each other, we implemented all the envisaged protocols (see Fig. 4), specifically BFCP, as it is currently specified in the IETF [5], and a brand new Conferencing Control Protocol we designed ourselves, whose features will be briefly described in the following. The interaction between the Focus and the Media Server led us to design an additional dedicated protocol for the remote control of the media processing functionality. More details upon this feature will be provided in the following sections.

As to BFCP, it has been implemented as a dynamic library, which has then been integrated into both client and server entities of the architecture. All the media management, manipulation and delivery have been bound to an event-driven mechanism, according to the directives coming from the floor control server.

Instead, considering that no agreement in the XCON WG

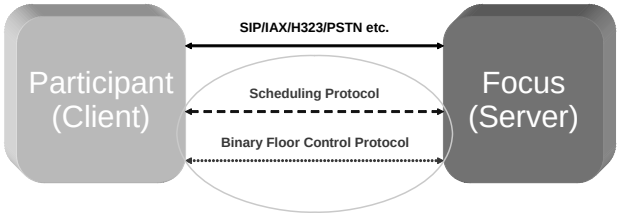

Fig. 4. New protocols implemented

had been reached about a specific Conference Control Protocol candidate at the time we were designing our architecture, we chose to develop a temporary, text-based, alternative solution ourselves, called *XCON Scheduler*. Such a Scheduler is capable to offer the basic functionality that the architecture is supposed to provide. Clients can use such protocol to dynamically manage conference creation as well as conference information.

A. Server side components

On the server side, we adopted Asterisk, a popular open source PBX which is constantly growing in popularity. The modular architecture behind Asterisk design allows it to be quite easily modified and enhanced, upon necessity. Specifically, we added to Asterisk the following new functionality:

- XCON-related identifiers, needed to manage conferences;
- Floor Control Server (FCS), by means of a dynamic library implementing the server-side behavior and policies of the BFCP;
- Scheduler Server, the server side component implementing the conference scheduling and management protocol;
- Video Mixer Client, the client side of the protocol implementing the interaction with the remote Video Mixer;
- Notification Service, to enable asynchronous events interception and triggering.

Most of these components have been realized as extensions to a conferencing facility already available as a module in Asterisk, called MeetMe. This facility acts as a set of configurable virtual "rooms" for channels that are attached to it, thus allowing users to access conferences by simply calling a predefined phone number, associated with a standard extension of Asterisk's dial-plan, independently from the clients signaling protocol. The addition of the above mentioned functionality allowed us to realize a fully-driven XCON compliant focus.

The addition of the *Scheduler* component to the "vanilla" MeetMe module allows for dynamic conference management in a user-friendly fashion: in fact, through this component clients are made able to dynamically (i.e. both in an active way, as in scheduling, and in a passive way, as in retrieving information) manipulate the conference objects and instances. Considering the dynamic nature of the framework with respect to policies, settings and scheduled conferences, all the required changes in the dial-plan, as well as dynamic reloading upon necessity, have been accomplished by adding the related functionality to the extended MeetMe module.

To allow video conferencing functionality, which was lacking in the base MeetMe module, we added a BFCP-moderated video support to MeetMe. Such support was accomplished by means of a brand new video mixer and transcoder we implemented from scratch, and which we called *Confiance VideoMixer*. The MeetMe application and the external VideoMixer communicate through a dedicated protocol, which is described in much more detail in the following subsection.

Since the XCON framework and data model define new identifiers (including the Conference URI and User ID [4]), as does the BFCP specification, the existing MeetMe data model has been enriched with the new required information.

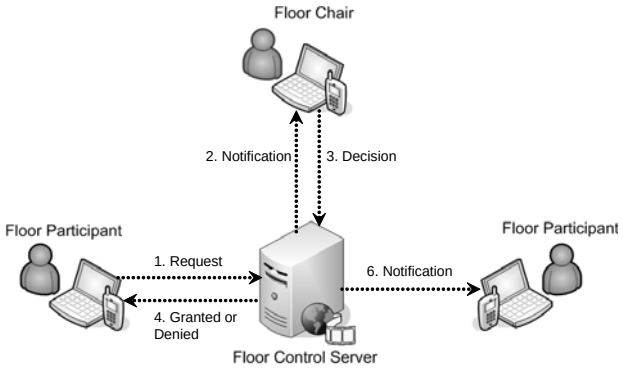

Fig. 5. The BFCP protocol in action

For what concerns BFCP, we had to implement the entire protocol, as well as its behavior which includes queues and state machines, from scratch. In order to achieve this, BFCP has been realized as a dynamic library, which is loaded at run time by the Asterisk server and comes into play whenever a resource is to be moderated. In fact, Asterisk, as the entity in charge of the business logic, also acts as the Floor Control Server of the architecture (see Fig. 5). The FCS functionality is involved every time a request is generated from a participant, asking for the right to access a specific resource (e.g. audio or video). As suggested by the picture, the FCS itself may or may not take any decision about incoming requests. In fact, while automated policies may be involved to take care of floor control in a more straightforward approach (e.g. to always accept or refuse incoming requests according to predefined policies), if they are not specified the FCS rather forwards floor requests to the designated floor chair, who is in charge of taking a decision that is accordingly notified to all the interested parties. As a transport method for BFCP messages, support for both TCP/BFCP and TCP/TLS/BFCP (as specified in [5]) has been implemented. Besides, since conference-aware participants, to take advantage of the BFCP functionality, need to know all the BFCP-related information of a conference, the focus needs means to provide her/him with such details. Apart from any out-of-band mechanism that could be exploited, the IETF has recently standardized a way [10] to encapsulate this information within the context of an SDP (Session Description Protocol) offer/answer. This functionality has been implemented as well in the module.

Coming to the Conference Control Protocol, in order to provide users with the capability of dynamically managing conferences, we designed and implemented a brand new protocol ourselves. This protocol, called *Scheduler*, has been conceived

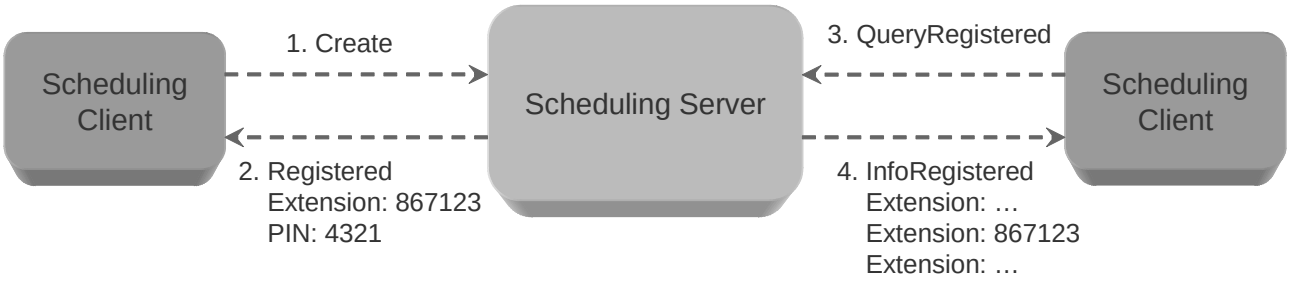

Fig. 6. The new protocol for conference scheduling

as a text-based protocol to be used by clients whenever a new conference instance is to be created/scheduled (see Fig. 6, left hand-side client), or the list of available/running conferences (see Fig. 6, right hand-side client) is to be provided.

Starting from this protocol, we also implemented a prototype Web Services-enabled wrapper to its functionality, and a proxy client that allows clients exploiting html browsers (e.g. for conference-unaware participants) to access and manage conference information. This kind of approach is the same the WG has recently started investigating for a new Conference Protocol Candidate, the Centralized Conferencing Manipulation Protocol [11].

Finally, we implemented a Notification Service by exploiting both existing solutions and customized modules. Besides reusing the already available Asterisk Manager Interface (which however only allows active notifications to passive listeners), we implemented a brand new protocol, which we called Dispatcher. This protocol is the base for a work we're carrying out in order to improve the scalability of the centralized conferencing framework, and is presented in detail in [12] and [13].

1) External VideoMixer: As already mentioned before, the existing conferencing module provided by Asterisk, which was the basis of our work, only supported audio natively. This obviously was a huge limitation in our framework, both for the user experience and for protocol research interest. In fact, having the possibility to involve moderation on different resources (i.e. not just on audio) provides us with more complex scenarios to deal with. Starting from these considerations, we first paved the way for a video support in the module by adding a basic video-switching functionality. The idea was basically to only allow one participant at a time to contribute to the video feed in a conference: this contribution would then be sent (or better, "switched") to all the other participants in the conference. This new functionality allowed us to start dealing with a video floor, thus introducing additional complexity in the BFCP interactions and offering interesting research ideas: in fact, the exclusive access to the video resource implied a strong role for the moderation protocol. However, a simple BFCP-moderated video-switching still couldn't satisfy us for many reasons. Apart from the already mentioned user experience, which could surely benefit from approaches like grid-based video layouts, video-switching, as the name suggests, is a simple blind forwarding of frames coming from a source to one or several destinations. This means that it is in no way concerned with content adaption, which could instead be needed when a conference involves participants making use of heterogeneous applications, devices and/or codecs. The most obvious example is two participants making use of different video codecs (e.g. H.261 and H.263): a blind forwarding would prevent both participants from watching the peer's contribution, if available. This led us to study the possibility of designing an ad-hoc media server which would act as a video multiplexer (for complex layouts involving more sources) and transcoder (to deal with video streams with different encodings and resolutions). To achieve this goal, we designed and implemented a custom video mixer, called *Confiance VideoMixer*. Considering our will to adhere to the separation of responsibilities principle, we chose this videomixer to be a remotely controllable media server. In this way, the conferencing module would only have to deal with the application logic (e.g. attaching participants to a mixed stream, specifying mix layouts, and so on), while the videomixer would process and manipulate the video streams according to directives sent by the module. As already mentioned, this approach is the same as the one currently investigated by the recently born Working Group in the IETF, called MEDIACTRL (Media Server Control), which will be further investigated in the final remarks. To design the client-server nature of the videomixer, we studied the requirements and potential issues of the already introduced separation of responsibilities: we thus specified a custom protocol allowing the conferencing module (or, more in general, any other entity managing the application logic) to control the manipulation of the participants' video streams. In the current implementation, the protocol allows for the per-user and per-conference customization of several aspects of the video processing, as layouts, transcoding, as well as the optional ability for participants to watch their own contribution in the mix they receive. All the directives the controller (in this case the conferencing module) sends to the videomixer are event-driven, and they make part of its application logic. Whenever a video-enabled participant joins a conference, its stream is redirected to the videomixer. By properly correlating the participant's identifiers associated with its own instances in

the controller and in the videomixer, the controller is then able to command different actions on the stream. BFCP moderation is one of the above mentioned events that can result in an action being requested by the controller: the video floor being granted to a participant would have the controller request the related participant's video stream to be included in the overall mix, just as the same floor being denied or revoked would result in an opposite request.

B. Client side components

On the client side, we chose an existing open source SIP softphone, Minisip, as the basis for our work. Most of the available configuration widgets used in Minisip have been appropriately modified in order to enable user-friendly support for XCON- and BFCP-related settings. Furthermore, brand new widgets have been created in order to allow users to take advantage of the functionality related to both conference scheduling and BFCP. As to conference scheduling, a widget has been implemented, which provides users with the needed conference scheduling functionality (see Fig. 7): as the figure shows, through such widget it becomes easy to either create a new conference, or retrieve the list of active XCON conferences, or join an existing XCON conference.

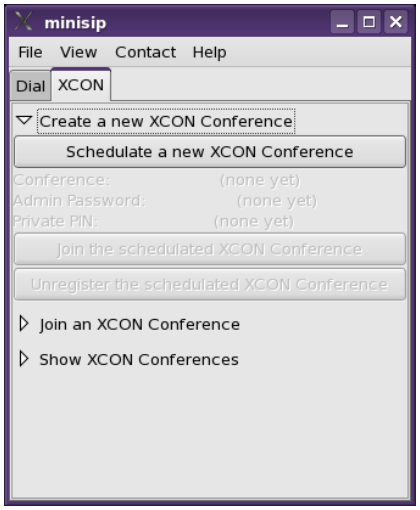

Fig. 7. XCON conference support in Minisip

Additionally, we implemented the client side BFCP behavior by designing new classes in Minisip. These classes act as wrappers to the dynamic library that has been introduced in the previous section. Then new widgets, associated with such classes, have been provided in order to enable users to:

- send BFCP messages to the BFCP server;
- interactively build BFCP floor requests in a user-friendly fashion, either in *participant* or in *chair* (i.e. with enhanced floor management functionality) mode;
- keep an up-to-date log of all the BFCP messages exchanged with the server (and optionally show each such message in further detail by simply clicking on the related entry in the history widget).

With respect to the role of the chair, we added ad-hoc interfaces in order to enable potential moderators to either manage floor requests issued by conference participants (an example of such interfaces is shown in Fig. 8), or build so-called *third-party* floor requests, i.e. requests generated by the chair on behalf of a different participant[1].

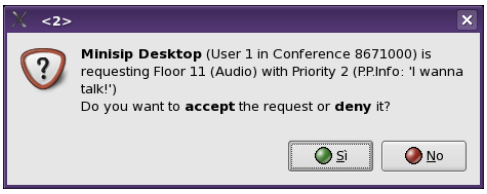

Fig. 8. Minisip: the role of the chair

To take advantage of the already mentioned negotiation of BFCP information within the context of the SDP offer/answer, we also added to Minisip the support for the encapsulation of BFCP information in SDP bodies. In this way, the BFCP is automatically exploited whenever a SIP INVITE (or reINVITE, in case the negotiation is involved in a subsequent moment) contains BFCP-related identifiers. Besides, the appropriate transport method for the BFCP communication with the FCS (i.e. TCP/BFCP or TCP/TLS/BFCP) is automatically chosen and exploited with respect to this SDP negotiation.

C. An example of client-server interaction

To provide the reader with a more detailed overview of the way the client-server interaction involves the introduced protocols, this section is devoted to presenting an example regarding a typical use case scenario. To ease the understanding of the sequence diagram (see Fig. 9), each protocol is represented with a different line style:

1) a participant (client in the scenario) contacts the Focus (the server), through the text-based scheduling protocol (dashed line), to ask for the list of currently active conferences, thus sending a *QueryConferences* message request with `Active` as argument;
2) the Focus processes the request and sends back to the participant (still through the scheduling protocol) an *InfoConferences* message, containing the list of all active conferences;
3) the participant reads the list and decides to join the active conference number 8671000: to join the conference, she/he calls the conference number – as if it were a standard phone number – using SIP (solid line) as the call signaling protocol, thus placing a call to the SIP URI 8671000*@Focus* (where Focus is the SIP domain, in this case the IP address of the P-CSCF);
4) the Focus receives the call and, according to the specified dialplan rules, routes it to the XCON-enabled MeetMe instance managing the conference with the same call number;
5) the XCON-enabled MeetMe instance managing the conference, through IVR (*Interactive Voice Response*), plays back a series of pre-recorded voice messages to welcome

[1]It is worth noting that such functionality is particularly interesting since it enables the chair to allow conference-unaware participants to take part to an XCON-enabled conference

the new user. It also warns the client about the fact that she/he is initially muted in the conference;

6) all the relevant BFCP information is encapsulated in an SDP body, and then sent back to the new user by means of a SIP re-INVITE;
7) once the client receives the re-INVITE and becomes aware of the needed BFCP set of data, she/he, using the BFCP (dotted line), decides to make a Floor Request to ask the Focus for the permission to talk;
8) the Focus, as Floor Control Server, answers the client by sending back a FloorRequestStatus BFCP message notifying that the request is currently pending. At the same time, the Floor Control Server forwards the message to the chair of the requested floor as well, to ask him to take a decision about the request.

From this point on, the BFCP transaction proceeds exactly as described before (see Fig. 5). Once the chair grants the floor, the client is un-muted and thus given the permission to talk until the floor is not willingly released by the client herself/himself or revoked by the chair. Since a floor is a logical object, all BFCP transactions will proceed in the same way, independently from the set of resources (be it audio or video, in the case of our platform) the related floor(s) could be associated with. In case the floor request involved a manipulation of a video request, a subsequent interaction between the conferencing module and the remote videomixer would take place through the dedicated protocol.

V. Related Work

The architecture we presented in this paper focuses on two main aspects: (i) compatibility with the IMS framework; (ii) capability to offer advanced functionality such as floor control, conference scheduling and management, etc. While there is a rich literature on each of the above points, when considered alone, to the best of our knowledge no integrated effort has been made to date which tries to provide a real architecture being both IMS compliant and capable to offer floor management functionality. This is mainly due to the fact that no agreed-upon solution has been so far designated in the various international standardization fora, with respect to some crucial points, such as the choice of the most suitable conferencing control protocol, as well as its integration in the general IMS architecture. Interestingly enough, a few works have already proposed to make a further step ahead, by moving from a centralized to a distributed perspective. This is the case, for example of [14], where the authors propose a model trying to extend the XCON approach to a distributed scenario. While this is currently out of the scope of the IETF, it does represent one of our primary goals for the next future, as it will be explained in the next section. On the IMS side, some efforts have already been devoted to the realization of IMS compliant testbeds, as in the case of [15], where the authors propose a testbed for multimedia services support based on the IMS specification.

Finally, several other works can be found in the literature, though based on superseded models such as those defined in the IETF SIPPING Working Group. This is the case, e.g. of [16] and [17].

VI. Conclusions and Future Work

In this paper we presented an actual implementation of an IMS-compliant architecture aimed at offering a video conferencing service with enhanced functionality, such as conference scheduling facilities and conference moderation, and focused on the interaction amongst the involved parties through the suite of dedicated protocols. The system we developed is both based on open source components, which have been appropriately enhanced in order to introduce support for the new required protocols and mechanisms, and on brand new software, which we realized to implement the state-of-the-art functionality. This allowed us to investigate many issues related to the envisaged tightly coupled, centralized conferencing model, including complex protocols interaction (e.g. the role of BFCP in media delivery), mechanism to enhance scalability (e.g. the separation of responsibilities between business logic and media manipulation for CPU-intensive functionality as video processing) and much more.

Nevertheless, a lot of challenging issues still remain and are to be faced. The scalability matter in particular, considering the centralized approach the XCON model takes, represents a very interesting field of research in many aspects and areas of the deployed architecture. For what concerns media manipulation, we have already presented the introduction of the VideoMixer as a separate entity from the Application Server. This VideoMixer, however, has only represented the seeds for a much more in depth work in this sense, which we have already carried on for some months at the time of writing. In fact, starting from the standardization efforts in the already mentioned MEDIACTRL WG, we have implemented a complete, modular, external Media Server in charge of all media-related manipulation and processing. This allows the Application Server to deal just with the signaling and control plane.

To further leverage the weight from the AS shoulders, we also already defined an architecture capable to realize a distributed conferencing system having strong reliability and scalability properties. Starting from the available centralized conferencing system, we have designed the overall architecture for distributed conferencing in terms of framework, data model and protocols definitions. The framework under definition has been called *DCON*, standing for *Distributed Conferencing*, but at the same time explicitly recalling the already standardized XCON model. So far, DCON has been implemented as a large scale evolution of the XCON framework. We have been proposing to deploy our architecture on top of a two-layer network topology, and many publications (including journals, conferences and IETF drafts) have been devoted in spreading our approach. The top layer is represented by an overlay network in which each node plays the role of the focus element of an XCON “island”. The lower layer, in turn, is characterized by a star topology (in which the central hub is represented by the focus element) and is fully compliant with the XCON specification. In the DCON scenario, communication among different islands (i.e. among the focus elements managing different islands) becomes of paramount importance since it enables to share information

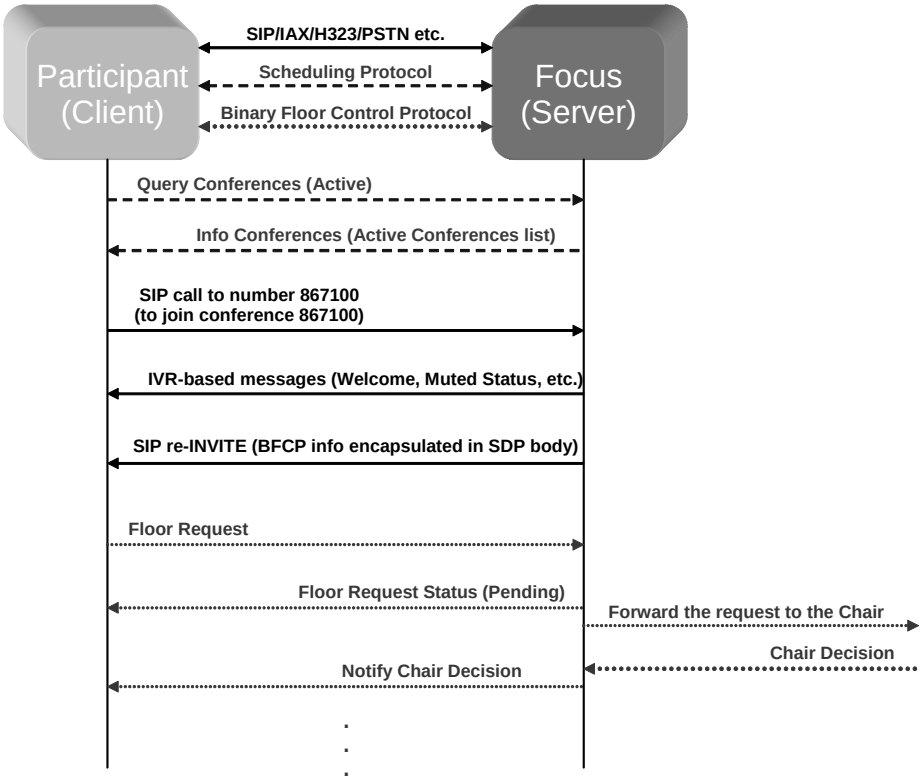

Fig. 9. An example of signaling between client and server

about the state of the available conferences, as well as about the participants involved in a distributed conference. To the purpose, we are investigating the possibility of adopting the so-called S2S (*Server to Server*) module of the XMPP (*Extensible Messaging and Presence Protocol*) protocol. XMPP has been standardized by the IETF as the candidate protocol to support instant messaging, e-presence and generic request-response services, and it looks to us as the ideal communication means among DCON focus entities. A prototype of the platform is already available (`http://dcon.sf.net/`) and currently provides distributed videoconferencing functionality.

ACKNOWLEDGMENTS

This work has been carried out with the financial support of the European projects NetQoS, OneLab and Content. Such projects are partially funded by the EU as part of the IST Programme, within the Sixth Framework Programme.

Alessandro Amirante received both his BSc and MSc Degree in Telecommunications Engineering from the University of Napoli "Federico II" in 2004 and 2007, respectively. He is currently a Ph.D. student in Computer Engineering and Systems at the Computer Science Department of University of Napoli "Federico II". His research interests primary fall in the field of networking, with special regard to Next Generation Network architectures and multimedia services over the Internet.

Tobia Castaldi received his degree in Telecommunications Engineering from the University of Napoli "Federico II", Italy, in 2006. He is currently a junior researcher at the Computer Science Department of the University of Napoli Federico II. The main topic of his research concerns real-time applications for the next-generation Internet with special regard to the IP Multimedia Subsystem (IMS) architecture and services.

Lorenzo Miniero received his degree in Computer Engineering from the University of Napoli "Federico II", Italy, in 2006. He is currently a Junior Researcher at the Computer Science Department of the same University. His research interests mostly focus on Next Generation Networks, network real-time applications, and communication protocols, with special emphasis on the related standardization efforts.

Simon Pietro Romano received the degree in Computer Engineering from the University of Napoli Federico II, Italy, in 1998. He obtained a PhD degree in Computer Networks in 2001. He is currently an Assistant Professor at the Computer Science Department of the University of Napoli. His research interests primarily fall in the field of networking, with special regard to QoS-enabled multimedia applications, network security and autonomic network management. He is currently involved in a number of research projects, whose main objective is the design and implementation of effective solutions for the provisioning of services with quality assurance over Premium IP networks. Simon Pietro Romano is member of both the IEEE Computer Society and the ACM.

REFERENCES

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, et al. SIP: Session Initiation Protocol. RFC3261, June 2002.

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

[3] M. Barnes, C. Boulton, and O. Levin. A Framework for Centralized Conferencing. draft-ietf-xcon-framework-10, November 2007.

[4] O. Novo, G. Camarillo, D. Morgan, and R. Even. Conference Information Data Model for Centralized Conferencing (XCON). draft-ietf-xcon-common-data-model-09, February 2008.

[5] G. Camarillo, J. Ott, and K. Drage. The Binary Floor Control Protocol (BFCP). RFC4582, November 2006.

[6] 3GPP. Conferencing using the IP Multimedia (IM) Core Network (CN) subsystem; Stage 3 (TS 24.147 7.1.0). Technical report, 3GPP, March 2006.

[7] OMA. Instant Messaging using SIMPLE Architecture. Technical report, OMA.

[8] OMA. Push to talk over Cellular (PoC) - Architecture. Technical report, OMA.

[9] 3GPP. IP multimedia subsystem; Stage 2, Technical Specification. Technical report, 3GPP, June 2006.

[10] G. Camarillo. Session Description Protocol (SDP) Format for Binary Floor Control Protocol (BFCP) Streams. RFC4583, November 2006.

[11] M. Barnes, C. Boulton, and H. Schulzrinne. Centralized Conferencing Manipulation Protocol. draft-barnes-xcon-ccmp-03, November 2007.

[12] A. Buono, S. Loreto, L. Miniero, and S. P. Romano. A Distributed IMS Enabled Conferencing Architecture on Top of a Standard Centralized Conferencing Framework. *IEEE Communications Magazine*, 45(3), March 2007.

[13] A. Amirante, T. Castaldi, L. Miniero, and S. P. Romano. Improving the scalability of an IMS-compliant conferencing framework through presence and event notification. In *Proceedings of the 1st International Conference on Principles, Systems and Applications of IP Telecommunications (IPTComm), New York City, NY*, 2007.

[14] Y. Cho, M. Jeong, J. Nah, W. Lee, and J. Park. Policy-Based Distributed Management Architecture for Large-Scale Enterprise Conferencing Service Using SIP. *IEEE Journal On Selected Areas In Communications*, 23:1934–1949, October 2005.

[15] T. Magedanz, D. Witaszek, and K. Knuettel. The IMS Playground @ Fokus An Open Testbed For Next Generation Network Multimedia Services. In *Proceedings of the First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities (TRIDENTCOM05)*, 2005.

[16] Z. Yang, M. Huadong, and J. Zhang. A Dynamic Scalable Service Model for SIP-based Video Conference. In *Proceedings of the 9th International Conference on Computer Supported Cooperative Work in Design*.

[17] A. Singh, P. Mahadevan, A. Acharya, and Z. Shae. Design and Implementation of SIP Network and Client Services. In *Proceedings of the 13th International Conference on Computer Communication and Networks (ICCCN), Chicago, IL*, 2004.