Third Party Control in SIP Conferencing

Ivaylo I. Atanasov, Pavlina H. Koleva, and Evelina N. Pencheva

Abstract – **The paper investigates capabilities of integrating third party applications control in control SIP (Session Initiation Protocol) conferencing. The capabilities of OSA (Open service access) application programming interface for conference call control are studied to identify which of the requirements for tightly coupled SIP conferences are met. Suggestion is made about new methods of OSA conference call control interfaces to support SIP conferencing requirements.**

Keywords – **IP-Multimedia Subsystem, Session Initiation Protocol, Open service Access, Conference call control**

I. INTRODUCTION

The IP- Multimedia Subsystem (IMS) of Next Generation Networks supports all kind of value-added services. In order to be able to implement future applications/end user services that are not yet known today, a highly flexible framework for services is required. Open Service Access (OSA) enables applications implementing the services to make use of network functionality through application programming interfaces (API). OSA provides the glue between applications and network functionality. In this way applications implementing the services become independent from the underlying network technology.

IMS services can reside either in the user's home network or in a third party location and are based on Session Initiation Protocol (SIP) [1]. The main components of the IMS, involved in SIP signaling, are the Call Session Control Functions (CSCF) [2]. The SIP servers with functionality of Call Session Control Functions perform a number of functions such as multimedia session control and address translation function. The Call Session Control Functions cooperates with Application Servers via the IP multimedia service control interface. The Application Servers where IMS services reside in might be:

- SIP Application Server which may influence and impact on the SIP session on behalf of the end systems, depending on the services.
- Intelligent network Application Server; the purpose of which is to host the Customized Applications for Mobile network Enhanced Logic (CAMEL) network
- OSA service capability server (OSA SCS) - which offers access to the IMS for the OSA Application Server in a standardized way for third parties.
- OSA Application Server which provides the service logic execution environment for client applications using the OSA API.

Ivaylo I. Atanasov, Pavlina H. Koleva and Evelina N. Pencheva are with the Faculty of Telecommunications, Technical University of Sofia, Kliment Ohridski 8, 1000 Sofia, Bulgaria, E-mail: iia@tu-sofia.bg, p_koleva@tu-sofia.bg, enp@tu-sofia.bg

Fig.1 shows the value-added service architecture over IMS.

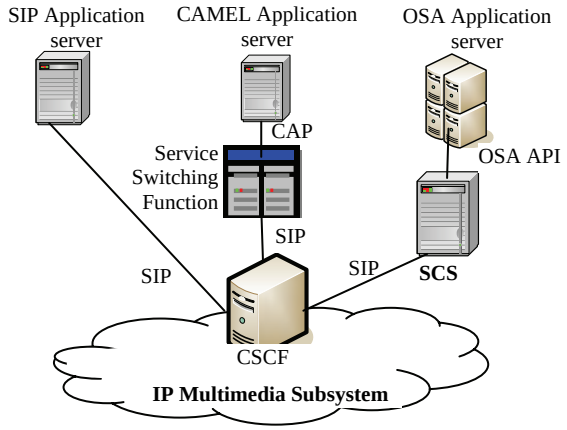

Fig.1 IMS service architecture

The OSA offers four standardized interfaces for call control. The Generic Call Control API defines simple call control interface which allows only setup of traditional two-party telephone calls [3]. The Multiparty Call Control API allows control of calls with zero or more parties and distinguishes between the call and its connections [4]. The Multimedia Call Control API deals with multimedia connections between parties [5]. The Conference Call Control API is for control of multimedia calls in which there exists the possibility of defining additional relationships between the parties [6]. The mapping of OSA Multiparty Call Control API onto SIP signaling is provided in [7]. There is no mapping between Multimedia Call Control API and Conference Call Control API onto SIP signaling.

In this paper we investigate the capabilities of OSA Conference Call Control API to support SIP conferencing. Considering high level requirements for tightly coupled SIP conferencing [8] we identify the way SIP conferencing requirements are supported by OSA and suggest extensions of Conference Call Control API for SIP conferencing requirements which are not supported.

II. SIP CONFERENCE FRAMEWORK

A tightly coupled SIP conference [9] is an association of SIP user agents (conference participants) with a central point, conference focus. The conference focus is a logical role and applies conference policy. The focus can be implemented either by a participant or by a separate application server. A dedicated conference server, in addition to the basic features, offers richer functionality including simultaneous conferences, large scalable conferences, reserved conferences, and managed conferences. A conferencing server can support any subset of the advanced conferencing functions. The conference URI is a SIP URI that identifies the focus of the conference.

The media graph of a SIP conference can be centralized, decentralized, or any combination of both, and potentially differ per media type. In the centralized case, the media sessions are established between the focus and each one of the participants. In the de-centralized case, the media graph is a mesh among the participants. The media processing can be performed either by the focus alone or by the participants.

A side-bar (subconference) is a conversation amongst a subset of the participants to which the remaining participants are not privy.

In case of OSA control of SIP conferences the conference focus is integrated with Conference Call Control Service Capability Feature (SCF). The SCF is abstraction of network functionality and is provided by Service Capability Server.

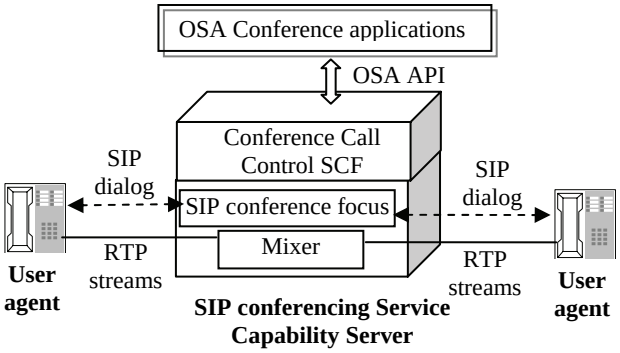

Fig.2 SIP conferencing OSA control architecture

Fig.2 shows a possible physical realization of basic functions. This is the classic "one box" solution where along with the focus functions are also implemented mixing functions. The mixer receives a set of media streams of the same type, and combines their media in a type-specific manner, redistributing the result to each participant. This includes media transported using RTP.

III. OSA Conference Call Control API

Most of the OSA interfaces follow a common structure.

The Conference Call Control API provides three interfaces at network side. The IpConfCallControlManager interface is the factory interface for creating conferences. The IpConfCall interface manages subconferences (side-bars). The IpSubConfCall interface provides grouping mechanism within conference.

There are also three interfaces at application side. The IpAppConfCallControlManager interface provides application with additional callbacks when a conference is created by the network. The IpAppConfCall interface allows applications to handle call responses and state reports. The IpAppSubConf-Call interface allows applications to handle call responses and state reports form a subconference.

In the next section we provide a study on OSA Conference Call Control API's support of SIP conferencing requirements, as shown in fig.2.

IV. OSA Conference Call Control and SIP Conferencing Requirements

A. Discovery phase

The discovery phase allow to reveal a location of SIP conferencing server, to determine if a referred SIP entity has focus capabilities and to obtain conference characteristics based on conference ID. In case of implementation shown in Fig.2, these requirements can be met by configuration means or by using proprietary conventions.

B. Conference creation

The OSA application can create a pre-arranged conference by invoking 'createConference()' method of IpConfCall-ControlManager interface. The Conference Call Control API provides methods for resource reservation and resource release to support pre-arranged conferences.

C. Conference termination

The IpConfCall interfaces inherits from IpMultiPartyCall interface its 'release()' method which requests the release of conference call object and associated objects. The OSA Conference Call Control API does not provide means for requesting a focus to revert a two-party conference to a basic SIP point-to-point session. A method of IpConfCall interfaces may be defined to request such transformation including the release of the associated conferencing resources.

D. Participant's manipulation

The OSA application can request from the conference focus to invite or to disconnect a participant by invoking 'create-AndRouteLegReq()' method and 'release()' method of IpSub-ConfCall interface inherited from IpMultiMediaCall interface.

The IpAppConfCall interface allows the application to handle parties entering and leaving the conference. Its method 'partyJoined()' indicates that a new party has joined the conference. This can be used in case the application implements conference policy to allow or reject a new participant. By invoking 'leaveMonitorReq()' method the application can request a notification when a party leaves the conference. The 'leaveMonitorRes()' method invoked on application indicates that a party has left the conference.

The OSA application can invite a user agent or a list of user agents to a particular active conference. Fig.3 shows an example of requesting a focus to add a new resource to a conference.

A conference participant can join the conference anonymously by announcing its presence but without disclosing identity. A so called passive participant can join a conference in a "hidden mode", without disclosure of presence. The OSA allows anonymous participation no means are available for application to distinguish anonymous and hidden modes.

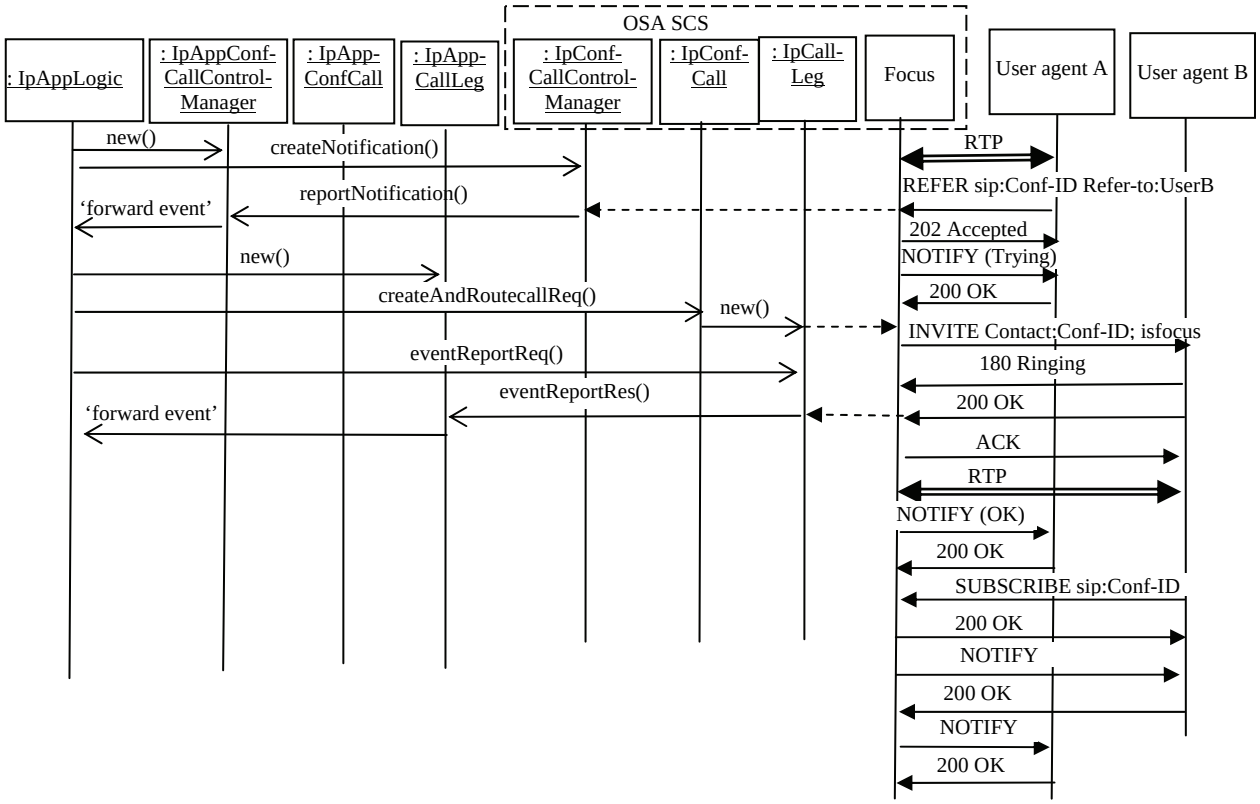

Fig.3 Requesting a Focus to Add a New Resource to a Conference

E. Conference state information

The conference state describes the conference in progress. This includes different conference aspects: participants' information (such as dialog identifiers and state), media sessions in progress (such as current stream contributing sources and encoding schemes), the current loudest speaker, the current chair, etc. Conference state is the latest conference snapshot triggered by changes in participants' state, conference policy changes, etc.

The list of conference participants can be received by OSA application by invoking 'getCallLegs()' method of IpConfCall interface inherited from IpMultiMediaCall interface. The application can invoke 'getConferenceAddress()' to receive address with which the conference can be addressed.

The application can register its interest in (selected) conference state changes including events like party joining and party leaving the conference.

The application uses createNotification() method to enable call notifications so that events can be sent. This is the first step an application has to do to get initial notifications of conference calls happening in the network. For example, when a user agent refers to a conference focus requesting a focus to invite another user agent to an active conference (see Fig.3), the application can be notified by invoking its 'reportNotification()' method.

To set, clear or change the criteria for the events concerning a particular participant in the conference, the application uses 'eventReportReq()' method of IpCallLeg interface. Reports that an event has occurred that was requested to be reported (for example an invited participant answers) are sent to application by invoking 'eventReportRes()' method of IpAppCallLeg interface.

If it is authorized, the OSA application can changed the conference policy in an ongoing conference by calling 'changeConferencePolicy()' method of IpSubConfCall interface. The OSA Conference Call Control API does not provide means for notifications when conference policy has changed. A 'confPolicyChanged()' method of IpAppConfCall interface might be defined for this purpose.

The SIP conference requests mean to express the minimum interval between receiving state change reports. To allow applications to specify the minimum state changes reporting interval a 'minReportingInterval()' method of IpConfCall-ControlManager interface might be defined.

Reserved conferences and ad hoc conferences may have a time limit. The conferencing system must inform timely participants when the limit is approaching and may allow the extension of the conference duration.

F. Focus role migration

OSA Conference call Control API does not provide means for delegating focus role by the current focus to another participant. It is not possible for applications to request a conference focus to transfer its role to different participant.

G. Side-bar conferencing

The application can create a new side-bar by invoking 'createSubConference()' method of IpConfCall interface.

IpSubConfCall interface provides methods for additional grouping within a conference. Participants (conference call legs) that are in the same side-bar have speech connection with each other. The application can create a new side-bar and move some participants to it by invoking 'splitSub-Conference()' method. To merge two side-bars the application uses 'mergeSubConference()' method. The application invokes 'moveCallleg()' method to move a participant from one side-bar to another side bar.

OSA Conference Call Control API does not provide means for reporting that a new side-bar is created. A 'subConferenceCreated()' method of IpAppConfCall interface might be defined.

V. OSA Support in Floor Control

Floor control enables applications or users to gain safe and mutually exclusive or non-exclusive input access to the shared object or resource. The floor is an individual temporary access or manipulation permission for a specific shared resource (or group of resources) [10]. Floor control is an optional feature for conferencing applications. SIP conferencing applications may also decide not to support this feature at all. Floor control may be used together with the conference policy control protocol or it may be used as an independent stand-alone protocol, e.g. with SIP.

Conference owner is a privileged user who controls the conference, creates floors, and assigns and deassigns floor chairs. The conference owner does not have to be a member in a conference. The OSA application in a role of conference owner can indicate which participant in the conference is the chair by invoking 'chairSelection()' method of IpSubConfCall interface. To inform the application about the chair selection requests from the network, 'chairSelection()' method of IpAppSubConfCall interface is used, and then the application can grant the requested.

Floor chair is a user (or an entity) who manages one floor (grants, denies, or revokes floor). The floor chair does not have to be a member in a conference. The OSA application in a role of floor chair is informed about the floor requests from the network by calling 'floorRequest()' method of IpAppSub-ConfCall interface. The application can grant the request by invoking the 'chairSelection()' method of IpSubConfCall interface. Using 'appointSpeaker()' method of IpSubConfCall interface, the application can indicate which of the participants in the conference has the floor, and the video of the speaker will be broadcast to the other parties. The application can call 'inspectVideo()' method of IpSubConfCall interface to select which video should be sent to the party that is currently selected as the chair. To cancel a previous 'inspectVideo()', the application calls 'inspectVideoCancel()' method, and then the chair will receive the broadcasted video.

VI. Conclusion

Research presented in this article shows that OSA Conference Call Control API supports most of the requirements for SIP conferencing. Using the API, OSA application can create and managed conferences, and group participants in sub-conferences. Still there are some SIP conferencing requirements that are not supported by OSA. Examples include lack of means to inform the application that conference policy has been changed, that sub-conference has been created, or about approach of conference time limit. There are no means in OSA to define minimum interval between changes report which prevents application from overload, and it is not possible to request hidden mode participation.

While the SIP based Application Server only runs services under some form of control by the IMS operator, the OSA Application Server integrates the screening functions of the OSA service capability server and offers the OSA interface to the OSA application server running third party applications. The standardized, extensible and scalable OSA interface allows for inclusion of new functionality in the network with a minimum impact on the applications using the OSA interface. This provides application developers with a power tool in designing new attractive multimedia services.

Acknowledgement

The research is partially funded by the project 08024ni-7, "Open Access to Telecommunication Resource Management in Next Generation Networks".

References

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

[2] 3GPP TS 23.218, IP Multimedia (IM) session handling; IM call model, v.7.7.1, 2007

[3] 3GPP TS 29.198-4-2 Open Service Access (OSA); Application Programming Interface (API) Part 4: Call Control Sub-part 2: Generic Call Control Service Capability Feature; v7.0.1, 2006

[4] 3GPP TS 29.198-4-3, Open Service Access (OSA); Application Programming Interface (API) Part 4: Call Control Sub-part 3:Multi-party Call Control Service Capability Feature; v7.0.1, 2006

[5] 3GPP TS 29.198-4-4, Open Service Access (OSA); Application Programming Interface (API) Part 4: Call Control Sub-part 4:Multimedia Call Control Service Capability Feature; v7.0.0, 2007

[6] 3GPP TS 29.198-04-05, Open Service Access (OSA); Application Programming Interface (API) Part 4: Call Control Sub-part 5:Conference Call Control Service Capability Feature; v7.0.0, 2007

[7] 3GPP TR 29.998-04-5, Open Service Access (OSA); Mapping for Open Service Access; Part 4: Call Control Service mapping; Subpart 4: Multiparty Call Control ISC; v7.0.0, 2007

[8] Levin O., R. Even, RFC 4245, "High-Level Requirements for Tightly Coupled SIP Conferencing", 2005

[9] J.Rosenberg, RFC 4353, "A Framework for Conferencing with the Session Initiation Protocol (SIP)", 2006

[10] Koskelainen P., J. Ott, H. Schulzrinne, X. Wu, RFC 4376, "Requirements for Floor Control Protocols", 2006