Research Journal of Applied Sciences, Engineering and Technology 4(22): 4914-4922, 2012
ISSN: 2040-7467

Submitted: June 11, 2012 Accepted: July 04, 2012 Published: November 15, 2012

Conference Control Design for a SIP-Web-Service Hybrid Multimedia Conference Framework

[1]Desheng Li and [2]Na Deng
[1]School of Science, Anhui Science and Technology University, Fengyang 233100, China
[2]School of Computer, Hubei University of Technology, Wuhan 430068, China

Abstract: Session Initiation Protocol (SIP) and Web service are the de facto standards for Voice over IP applications and distributed system**,** respectively. Although they do not be used to define multimedia conferencing directly and completely, many proposals, drafts and papers suggest extensions and solutions to this essential service. However, as the spring up of the Web service standards and technology, some traditional communication systems should be restructured and modified, including the multimedia conference framework based on SIP and Web service. In this study, a hybrid multimedia conference framework is first propose, which consist of low latency services, primitive and composite conference services. Under this framework, a multimedia conference can be organized and controlled conveniently with the property of loosely coupling and extensibility. Then the primary components of a conference control are introduced followed by the detailed design exemplified by some typical scenarios, which is compared to other possible design approaches. Finally, a specification of orchestration implementation which realizes the synthesis of the conference control functions is introduced to illustrate the feasibility of this design.

Keywords: Conference control, session initiation protocol, web service

INTRODUCTION

Conference control, one of the core modules of multimedia conferencing system, has been an area of intensive research over the past years but widely accepted robust and scalable solutions and standards are still immature or even lacking. It is required not just by traditional voice and video conferencing, but also for multiparty network gaming.

Problems relevant to conference control have been studied extensively over the years. Among them, most of the earlier work discusses only the floor control aspects of conference control, such as in Koponen (2008) and Malpani *et al*. (1997). Furthermore, most researches put their focuses on multicast conferences (Schubert *et al*., 1998; Sisalem *et al*., 1998); however, in the absence of a widely available Internet multicast service, we will consider the currently more common centralized conferencing architecture, especially using Sip stack and Web Service technology.

On the other hands, for a long time, the whole solution of conference control mechanism is always application and protocol independent. For instance, there exists a remarkable lag about the designs of conference control between SIP (Rosenberg *et al*., 2002) and H.323 (ITU, 1998a) architectures, also between SIP and H.248 (Taylor, 2000) in the same system. However, standardization efforts on these protocols have met with limited success. Concretely, the main drawbacks of the conference control in the existing architectures can be summarized as follows:

- In H.323, the inefficient T.124 (ITU, 1998b) database replication protocol is less scalable and lack some necessary key features. Besides, the concepts of data and audio/visual conversation is different that adding considerable complexity. In addition, they are based on binary ASN.1 formats instead of the more Internet-friendly text protocols used in current Internet protocols such as SIP and HTTP.
- In SIP, although it can initiate such conferences, allow users to join and leave conferences and make it possible for a participant to ask a third party to join. However, SIP by itself does not offer configurable control functions of conference, such as management of policies, participant access lists, floor control, user privilege levels and so forth. Moreover, as the booming of SOA and Web service, a requirement of converging of communicational services and other data service is necessary and also urgent.

The contribution of this study is the presentation of the hybrid architecture of the multimedia conference by SIP and Web servicee and the design route of the

Corresponding Author: Desheng Li, School of Science, Anhui Science and Technology University, Fengyang 233100, China

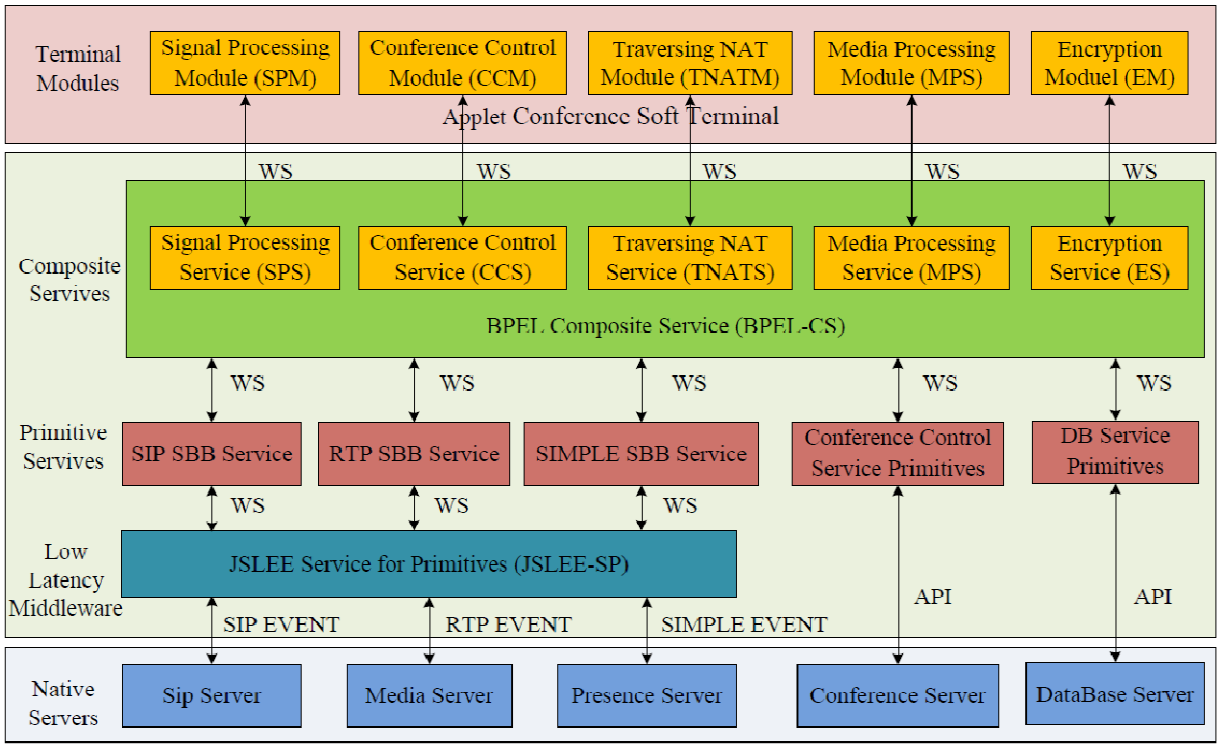

Fig. 1: Architecture of Sip-Web-Service hybrid multimedia conference

conference control which origins from our previous practice for BPEL based multimedia conference Web services orchestration as in Cheng *et al.* (2008), which can enable the top-down realization of SOC through coordination of conferencing communication Web services

ARCHITECTURE OF SIP-WEB-SERVICE HYBRID MULTIMEDIA CONFERENCE

In this study, we will introduce the whole architecture of our prototypical multimedia conference system using the hybrid approach of SIP and Web service. Figure 1 clarifies the components of the framework and highlights the three-tired hierarchical structure composed by the native servers, Web services and the terminal modules. The flexibility of our framework comes from the encapsulation to Web service from atomic operations of several native services in different languages even platforms and unification of invoking for these services

Native servers: The native servers include the main the communication and data service server, i.e., SIP server, Media server, presence server, conference server and database server, in which some are the legacy services in various programming languages. Due to the incompatibility of this server in protocol and language, the directly combine of them is next To Impossible. So, we need a high-leveled platform, i.e., Web service composition to integrate these services in a common interface and standards.

Although, the most communicational services can be achieved through intelligent manipulations in form of the SIP dialogs, but not all delivered through SIP imply just manipulation of SIP dialogs. In fact, there are a lot of applications that also imply some processing at the media level. Take for instance, a voice-mail service that requires the voice mail to play an announcement and record a message. Or think about services that require user input in the form of DTMF. Also, conferencing services imply specific media processing so as to enable the mixing of different streams. All these services have in common the fact that, in addition to the manipulation of the SIP dialogs (i.e., manipulation of the signaling); they also involve some specific processing at the media level. Media-processing functions include, but are not limited to:

- Play media, such as speech or video
- Record media, such as speech or video
- Prompt and collect media info from the user
- Mixing media streams
- Convert text to speech
- Speech recognition
- Transcoding media

Table 1: Third-party components

Component	Name	Version	Provider
SIP stack	Nist-sip	1.2	NIST
	JainSipApi	1.1	Sun
SDP stack	Nist-sdp	1.0	NIST
JMF stack	Jmf	2.1.1.e	IBM and sun
	Jmdaud	2.1.1.e	IBM and sun
Codec	Jmg723	2.1.1.e	IBM and sun
	Jmh263enc	2.1.1.e	IBM and sun

In our prototype, the Brekeke SIP server 2.4.8 (Brekeke, 2010) and Radisys (Convedia) CMS-9000 media server (Radisys, 2010); while the presence server is a plug-in hosts on the Brekeke SIP server. The conference server is a standalone server in Java and the database server is an application of MySQL.

Web service composition: As illustrated in Fig. 1, the layer of service orchestration comprises low latency middleware services, primitive services and the composite services. This layer is based on Apache (ServiceMix, 2006), which is an open source ESB (Enterprise Service Bus) that combines the functionality of a SOA and an Event Driven Architecture (EDA) to create an agile application that is to allow components and services to be integrated in a vendor independent way, allowing users and vendors to plug and play.

As the SIP service is not a standard Web service, so it needs to be converted into a true one. Firstly, we use the JAIN SLEE (Lim *et al*., 2004) to encapsulate the native SIP service, which, in this case, does not provide a functional API, but just the interface to a container for carrier-grade telecommunication applications. Thus, it must be combined with other functional APIs that provide the access to the protocol functionality, for example, JAIN SIP (Jain, 2010; SIMPLE 2010). The JAIN SLEE container represents a horizontal software layer that can accommodate almost any type of network protocols, thus being the natural choice for building applications that require both legacy signaling protocols-such as IN (Intelligent Network) protocols-as well as newer Internet-based protocols such as SIP.

Terminal: The terminal has the corresponding functional modules to the server side ones. The light boxes represent application components and the dark boxes refer to components in the underlying JAIN SIP, JAIN SDP, or JMF implementation provided by third parties. Table 1 summarizes the third-party components that our application is using.

Components of conference control: In this part, we will introduce an abstract framework of conference control of soft terminal based on the requirements and the whole conference system architecture as in the literature (Koskelainen *et al*., 2002). The control of a multimedia conference, i.e., Conference Control (CC), is the kernel of a conference system, which should be performed under the cooperation of both server and client sides. In the server side, The Conference Control Service (CCS) offers the conferencing services to the members. In the client side, the Conference Control Module (CCM) here is a soft terminal application that may or may not be co-located at the server.

In brief, the CC provides for high level data manipulation and state retrieval from the Web Service Primitive of Conference Operation. It allows us to create/delete/modify conferences, add/delete users, add/delete/modify media, set participants on mute, alter the gain media streams, assign roles to participants, create sidebars and so forth. In our SIP-Web-Service hybrid multimedia conference framework, we does not import some specific conference/media control protocol to add the complexity, such as BFCP (Camarillo *et al*., 2006), H.248 (Taylor, 2000).

The CC includes three principal components, floor control, user management, distribution and session management as illustrated in the latter parts. Apart from these components, conferencing requires also standard participant operations such as joining and leaving a conference. Current Internet session control protocols already provide most of the services needed for participant operations. For example, SIP can be used to initiate, modify and terminate sessions or transfer members. Standard SIP mechanism such as digest authentication also enforce user authentication. The Session Description Protocol (SDP) (Handley *et al*., 2006) allows participants to negotiate media parameters. But some advanced functions such as media conversation session control and floor control are difficult or impossible to be realized only with the above protocols. So the hybrid architecture of Web service and Sip shows convenient and powerful with the diversity of native services.

User management: The user management sub-component governs the participants' right to access to a conference, such as defining the user right groups, allowing and denying definitions for the join attempts and so forth. Generally, user management supports real-time decisions during the processing conference. Unresolved join attempts are kept in a table until the moderator, notified by the system, sends a request making a decision. In addition, it also supports a mass-invitation feature, in which a user may ask the conference server to invite several users at once into the conference.

Session management: The session management component takes responsibility for conference wide media configuration. For a multimedia conference system, session managements of signaling and media conversation are both important. Although SIP can control the sessions between one user and the conference server, but it can hardly change the

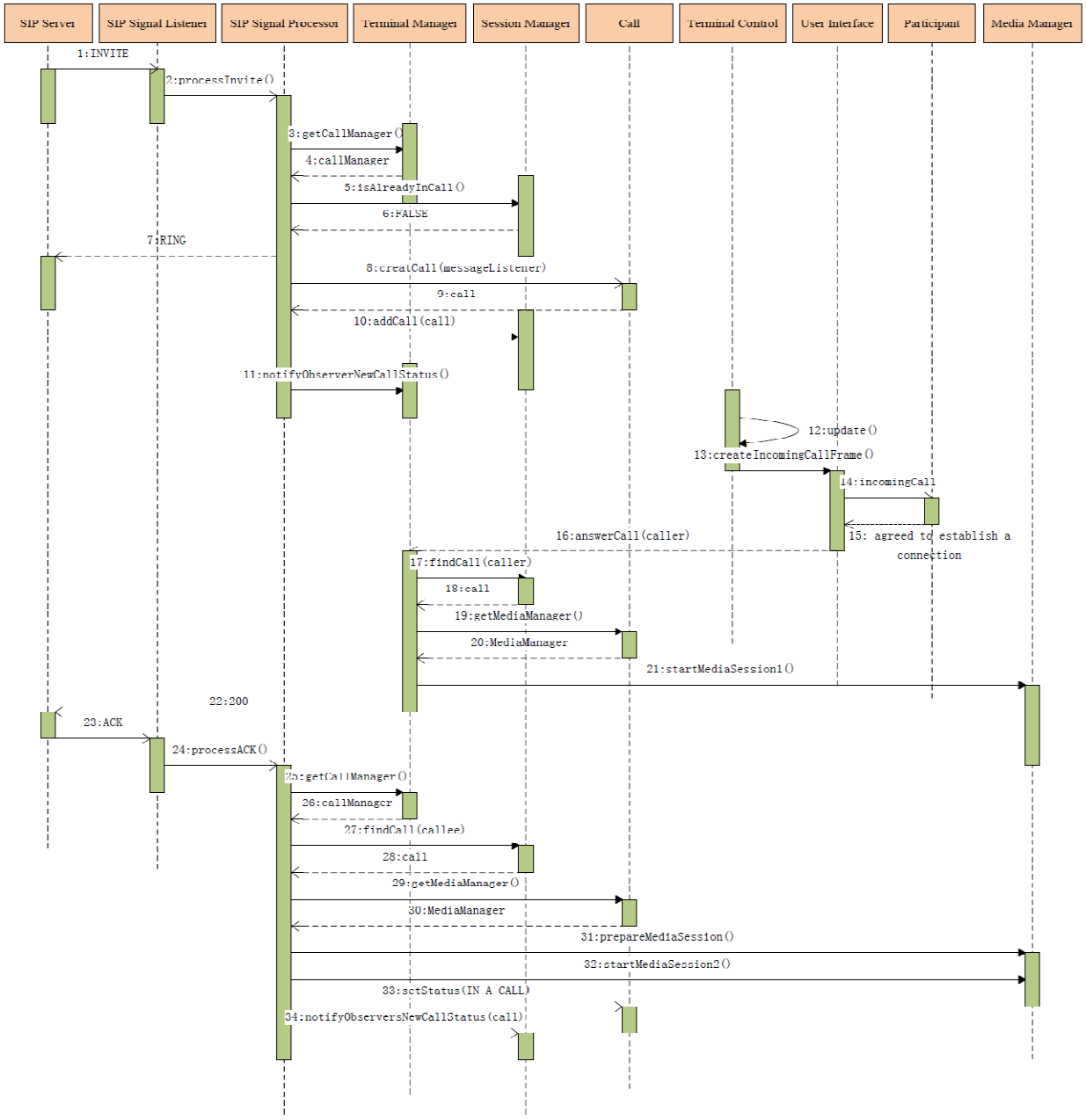

Fig. 2: Signaling and primitive service procedures for JCU

conference state or the conference media settings at both the server and client sides. For instance, deleting media from a conference only using SIP is difficult: if a participant removes the corresponding media line from the session description and sends a "re-INVITE" to the conference server, it is not clear whether the participant would like to personally not receive that media session or would like to request that the server stop replicating that particularly media session for all participants. And Vice versa: when a terminal changing media sessions, it is impossible to distinguish whether the participant wants to restrict media codec choices.

Floor control: Floor control takes responsibility for the management of the access to shared resources, for instance, the determination of who in the audience has the right to talk, who are not allowed to speak unless they have the floor and so forth. Concretely, if a participant wants to speak, he/she needs to make floor request to the floor chair (a management entity), which will grants or deny the request and inform the floor participant about his/her status/position in the floor's queue.

However, unfortunately, the floor control was not supported directly either in the architecture for basic

media services or in the SIP conferencing framework. For this reason, we implement it with the help of composite Web services integrated by the primitive services from the SIP server, media sever, presence server and conference server.

Distribution: The distribution subcomponent defines how the conference can be distributed across conference servers. But in our prototype and framework, we does not consider the distributed topology of conference servers to avoid the complexity, so in this paper, this function is not mentioned and may become an interest in our future work.

TYPICAL SCENARIOS

Some typical scenarios of conference controlling to illustrate how to combine the Sip signals and the primitive Web services to work seamlessly for the control design of conference. For clarity, we write an operation in a Web service in a form of traditional function.

Join Conference Unit (JCU)*:* The detailed interaction sequences for signaling and primitive service procedures for JCU are given in the Fig. 2. First, the SIP signal listener receives the "INVITE" message from the SIP server and calls the SIP signal processor to process it. Next, the SIP signal processor invokes get Call Manager () to get the handle of the session in the session manager and also judge that whether exists any on-going communication. If the session manager returns the session state that no session is active currently, then it informs the server "180 RING". After that, a new session will be established by signal processor and return its handle, which would be added into the list of the session manager. Next, the change of the session state will be informed to listener. Meanwhile, the terminal controller executes the update () method to check the timestamp, here it can detected the change. Also, the terminal controller creates a new inform dialog of new conference to remind the participant the arrival of a new call. If the user agree to establish the connection of the session, an answer Call () method would be invoke to reply this call. Then, the terminal manager calls the find Call () of the session manager to probe the handle of current session. When the handle is returned by the session manager, a handle of media manager will be acquired from the session object and returned. Afterwards, the start Media Session1 () of the media manager is invoked to listen the receive port of media to make sure that the probe package from the media server could be received errorless. Otherwise, if the port does not be opened in time (may after receive of "ACK"), then the media server may consider that the client could not be able to establish a connection and may end the communication with the terminal actively. Next, it sends "200 OK" to the SIP server while the client sends the "200 OK" repeatedly until the server receives the returned "ACK". When the SIP signal processor receives the "ACK" message from the SIP server, it informs the SIP signal processor to deal with the message by means of the invoking of get Call Manager () to obtain the handle of the session manager. According to it, another handle of media manager would be also obtained from the current session. Then, the prepare Media Session () of the media manager is invoked to get the receiving and sending media URIs of the remoted media server, the local URIs and the coding/encoding formats after negotiation. Meanwhile, the start Media Session2 () of the media manager is also invoked to set the state of the call "IN_A_CALL" and informs the session manager the fact of the change in the session state. By the above interactions of signals and services, the user could join the conference and establish the media connection successfully.

Remark: The "JOIN" header introduced in Mahy (2004) can be used inside an "INVITE" request in order to cause the joining of an existing SIP dialog with a new SIP dialog. This call control primitive enables services such as "Barge In," "Call Center Monitoring," and others. So a candidate design method is using "REFER" message and "JOIN" header sent by UA to tell the focus it would join the conference. For the consideration of compatibility of a mount of existing simple conference Web service primitives and the popular treatment of SIP protocol stack, we don't adopt it into our design. Moreover, compared with this approach, our design shows more loosely coupled and compatibility.

Quit Conference Actively (QCAU): A participant in a conference can quit it in two ways: one is quit it actively, called Quit Conference Actively Unit (QCAU), i.e., applying termination of its own session by sending a BYE message to the Sip server; the other is quit conference passively, called Quit Conference Passively Unit (QCPU), that is when a conference ends or the chairman executes operation of rescinding the participant status of this client, the Sip server then send BYE message to the client and ends the conference completely.

As shown in the Fig. 3, the execution message sequence of both Sip signaling and primitive Web service invoking of QCAU. First, when the UI detects a behavior of pressing "Quit Conference" button in its own approach service "control Button Action Performed (evt)", it immediately invokes the end Call (calleeURI) service primitive of the terminal manager. Next, the terminal manager invokes the find Call (calleeURI) of session manager to search the current processing session with the callee's URI. If it found, a handle of current session is returned rather than

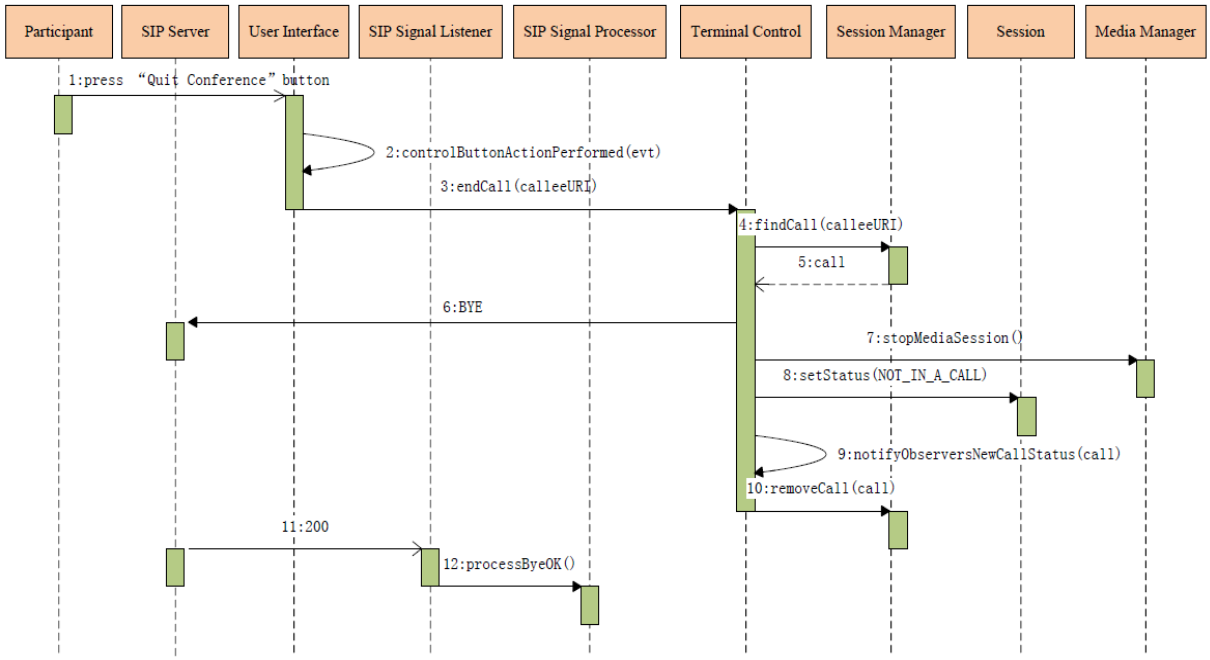

Fig. 3: Message sequence of signaling and service primitive invoking for QCAU

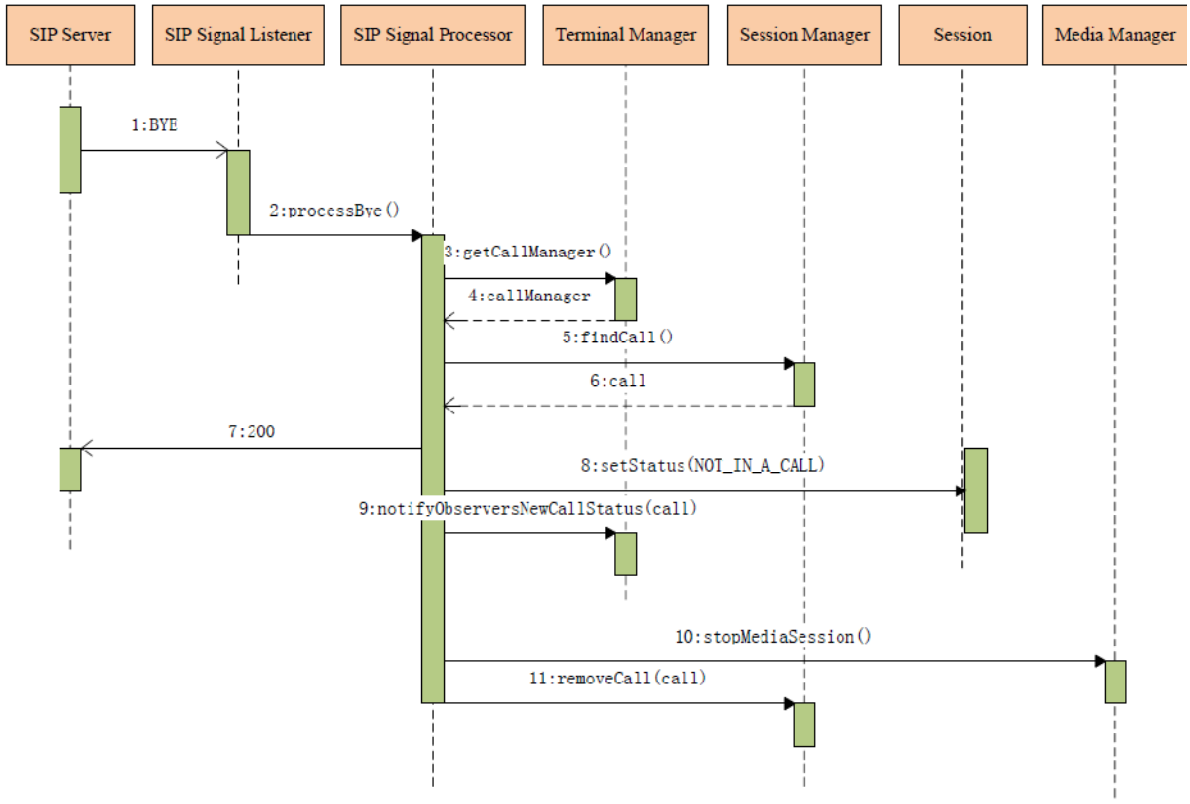

Fig. 4: Signaling and primitive service procedures for QCPU

"null". Meanwhile, the terminal manager also send a "BYE" signal to the Sip server, invoke the stop Media Session () of media manager to release resources, set the status of this session "NOT_IN_A_CALL", notify the observer the change of session status and remove the current session from the list of session manager.

Afterwards, the Sip server then sending 200 OK" to the Sip signal listener that can called the service process Bye OK () of the Sip signal processor to handle this message.

Remark: The alternative approach for leave the conference is the method using "REFER" Sip message, which is defined in Sparks (2003) that allows a User Agent to ask another User Agent to send a request on its behalf. It also provides a mechanism allowing the party sending the REFER to be notified of the outcome of the referenced request. However, as the lack of enough support in common SIP stack, such as JAIN-SIP, so for the consideration of compatibility, we don't import this signal into our design. The design for the next case follows this treatment.

Quit Conference Passively Unit (QCPU)*:* Fig. 4 shows the signaling and primitive service procedure of QCPU in which the conference participant quit the conference passively. First, on a BYE message from Sip server received by a Sip signal listener, the Sip signal listener invokes the primitive service process Bye () to process this signal. Then the Sip signal manager invokes the get Call Manager () of terminal manager and also get the handle of session manager. Terminal manager use find Call () of session manager to search the current session in active. If the result is not null, then return the current handle of session.

Otherwise, the Sip signal processor should do some operations. The first thing is to reply "200 OK" to the Sip Server, which subsequently reset the state of the found session "NOT_IN_A_CALL". Then it should

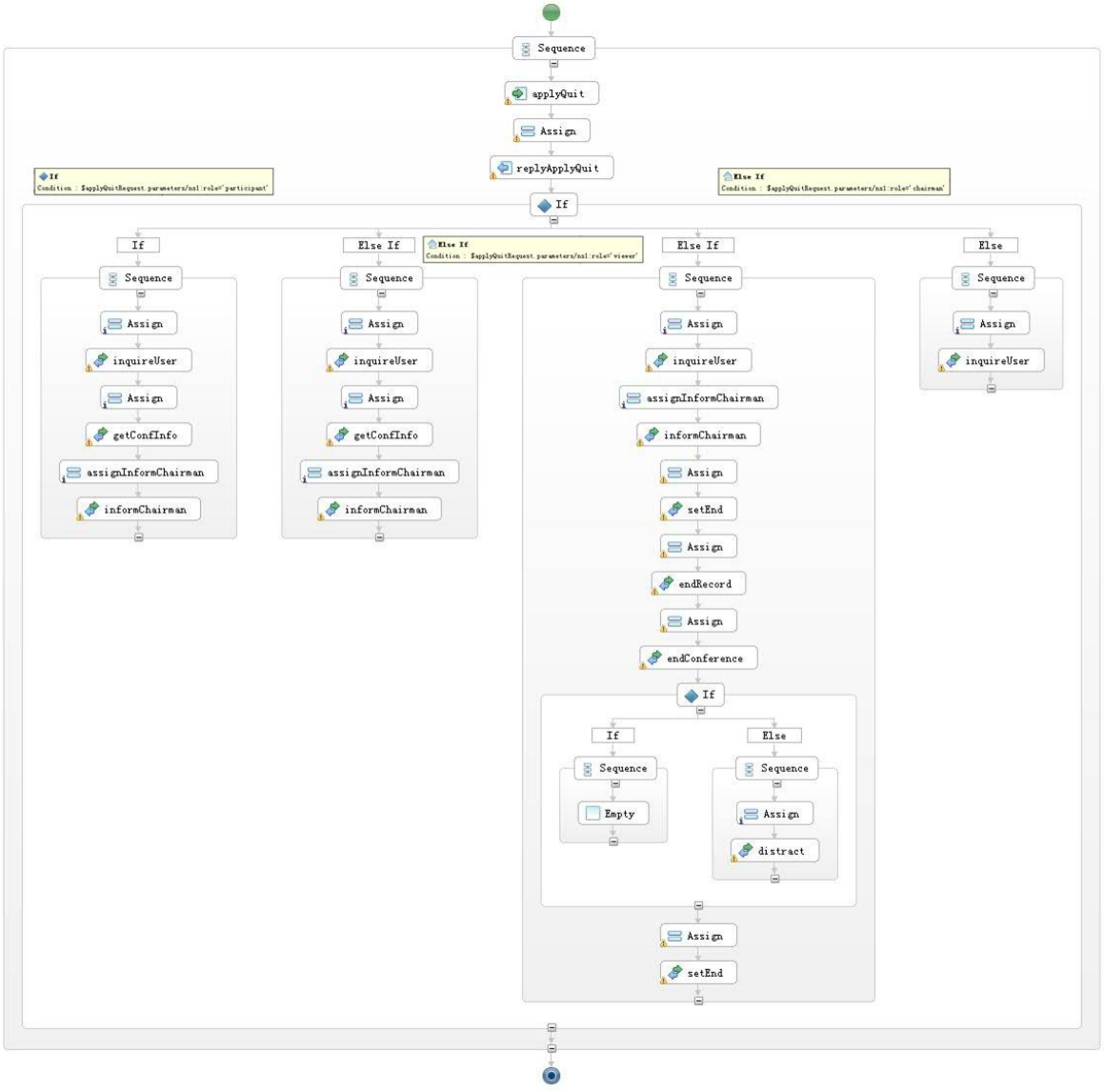

Fig. 5: BPEL flow of the apply join () composite service

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions ……>
<plnk:partnerLinkType name="applyQuit" ……>
  <plnk:role name="myRole" portType="impl:ApplyQuit"/>
</plnk:partnerLinkType>
<wsdl:types>
  <schema .....>
    <element name="applyQuit">
      <complexType>
        <sequence>
          <element name="confID" type="xsd:int"/>
          <element name="userID" type="xsd:int"/>
          <element name="role" type="xsd:string"/>
          <element name="clientType" type="xsd:string"/>
        </sequence>
      </complexType>
    </element>
    <element name="applyQuitResponse">
      <complexType>
        <sequence>
          <element maxOccurs="unbounded"
            name="applyQuitReturn" type="xsd:string"/>
        </sequence>
      </complexType>
    </element>
  </schema>
</wsdl:types>
<wsdl:message name="applyQuitResponse">
  <wsdl:part name="parameters"
            element="impl:applyQuitResponse"/>
</wsdl:message>
<wsdl:message name="applyQuitRequest">
  <wsdl:part name="parameters" element="impl:applyQuit"/>
</wsdl:message>
```

```
<wsdl:portType name="ApplyQuit">
  <wsdl:operation name="applyQuit">
    <wsdl:input name="applyQuitRequest"
              message="impl:applyQuitRequest"/>
    <wsdl:output name="applyQuitResponse"
              message="impl:applyQuitResponse"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="ApplyQuitSoapBinding"
                          type="impl:ApplyQuit">
  <wsdlsoap:binding …… />
  <wsdl:operation name="applyQuit">
    <wsdlsoap:operation ……"/>
    <wsdl:input name="applyQuitRequest">
      <wsdlsoap:body ……/>
    </wsdl:input>
    <wsdl:output name="applyQuitResponse">
      <wsdlsoap:body ……/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
<wsdl:service name="ApplyQuitService">
  <wsdl:port name="ApplyQuit"
            binding="impl:ApplyQuitSoapBinding">
    <wsdlsoap:address
      xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl
                          /soap/"
      location="http://localhost:8080/conferenceService/
                          services/ApplyQuit"/>
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Fig. 6: WSDL specification of the apply join ()

invoke notify Observers New Call Status (call) of the terminal manager and notice the observer the fact of change of session state. Meanwhile, the stop Media Session () service of media manager should be called to release the resources related to the using media. The final operation is to delete the session item from the active session list from the session manager.

The BPEL Composite Service Implementation: The BPEL (OASIS, 2007) is also the de facto standard for Web service orchestration like the SIP in VOIP. In our design of the conference system, it is used to integrate the various primitive services into some coarse-grained composite services.

Figure 5 shows the BPEL executing flow chart of the apply Quit () service of the conference control which is invoked after an input message of "apply Quit" from the terminal and returns a message of "reply apply Quit" asynchronously. From it, we can charily see that different roles, i.e., participant, viewer, chairman and others, have their own executing paths due to different limits of authority. The main functional primitive services include inquire User (), getConfInfo (), inform Chairman (), set End(), end Record (), end Conference () and distract (), which are invoked from different servers to implement this orchestration cooperatively. Especially, if the role is the chairman, some internal affairs services are fired to end the conference and dispose the resources in use.

As shown in Fig. 6, the WSDL (W3C, 2001) of apply Quit () service exposes the external interface to be invoked and some necessary message types. Concretely, the apply Quit element is composed by confide, userID, role and client Type, while apply Quit Response only with the string value of apply Quit Return. Finally, this kind of service is bind by SOAP message on the location of (http://localhost:8080/ conferenceService/services/ApplyQuit"/) for the invoking of other services.

CONCLUSION

Service-Oriented Communication (SOC) is a new trend in the industry to enable communication through a Service-Oriented Architecture (SOA) and thereby package communications as services. Based on the hybrid multimedia conference architecture of SIP and Web service, flexibility, loosely coupling could be

obtained rather than relying on some extended messages on the existing protocol. In the point view of this paper, a unified treatment to the messages of SIP signals and other Web services which brings the extensibility differs from other technique routes. Moreover, from the given scenarios, we illustrate the detailed message sequences among the terminal UI, SIP server, conference server and media server, which show the feasibility and clearness of the whole message flow. Finally, an orchestration in form of BPEL and WSDL which realizes the synthesis of the conference control functions is introduced to implement the design.

ACKNOWLEDGMENT

This study is supported by the Talent Introduction Special Fund of Anhui Science and Technology University (Grant No. ZRC2011304).

REFERENCES

Brekeke, 2010. Brekeke SIP Server-SIP Proxy, Registrar Server. Retrieved from: http: // www. brekeke. com/sip/.

Camarillo, G., J. Ott and K. Drage, 2006. RFC-4582, the Binary Floor Control Protocol (BFCP). pp: 1-65, Retrieved from: http :// tools. ietf. org/html /rfc 4582.

Cheng, B., J. Guo, X. Lin and J. Chen, 2008. A preliminary practice for BPEL based multimedia conference web services orchestration. Proceeding of the 4th Advanced International Conference on Telecommunications, pp: 321-326.

Handley, M., V. Jacobson and C. Perkins, 2006. RFC-4566, SDP: Session Description Protocol. University of Glasgow, UK.

ITU, 1998a. Packet based Multimedia Communication Systems. Recommendation H. 323, Tele Communication Standardization Sector of ITU, Switzerland.

ITU, 1998b. Generic Conference Control. Recommendation T.124, Telecommunication Standardization Sector of ITU, Switzerland.

Jain, S.I.P., 2010. Open Source Project. Retrieved from: https://jain-sip.dev.java.net/.

Koponen, A.H., 2008. A floor control server in a distributed conference service. M.S. Thesis, University of Helsinki, Helsinki.

Koskelainen, P., H. Schulzrinne and X. Wu, 2002. A SIP-Based Conference Control Frame Work. NOSSDAV'02, USA.

Lim, S.B. and D. Ferry, 2004. JAIN SLEE 1.0 Specification, Final Release. Retrieved from: http://www.jcp.org/en.jsr/detail?id=22.

Mahy, R. and D. Petrie, 2004. RFC 3911 the Session Initiation Protocol (SIP). Join Header, pp: 1-17, Retrieved from: http://tools.ietf.org/html/rfc3911.

Malpani, R. and L.A. Rowe, 1997. Floor control for large-scale mbone seminars. Proceeding of ACM Multimedia, USA.

OASIS, 2007. Web Services Business Process Execution Language Version 2.0 (BPEL 2.0). Retrieved from: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

Radisys, 2010. CMS-9000 Media Server. Retrieved from: http://www. radisys.com/Products/Media-Servers.html.

Rosenberg, J., H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, *et al*., 2002. RFC-3261 SIP: Session Initiation Protocol, IETF. Retrieved from: http://www.ietf.org/rfc/rfc3261.txt.

Schubert, I., D. Sisalem and H. Schulzrinne, 1998. A session floor control scheme. Proceeding of International Conference on Telecommunications, Greece ServiceMix, 2006. Open Source Project. Retrieved from: http://servicemix.Org.

Simple, 2010. The Session Initiation Protocol for Instant Messaging and Presence Leveraging Extensions. Retrieved from: http:// datatracker. ietf. org/wg /simple /charter/

Sisalem, D. and H. Schulzrinne, 1998. The Multimedia Internet Terminal (MInT). Telecommun. Syst., 9(3-4): 423-444.

Sparks, R., 2003. RFC 3515 The Session Initiation Protocol (SIP). Refer Method, pp: 23, Retrieved from: http://www.testingtech.com/solutions/ttsuite-sip.php.

Taylor, T., 2000. H. 248: A new standard for media gateway control. IEEE Commun. Mag., 38(10): 124-132.

W3C, 2001. Web Services Description Language (WSDL) 1.1. Retrieved from: http:// www. w3. org/TR/wsdl.