Integration of SIP VoIP and Messaging with the AccessGrid and H.323 Systems

Wenjun Wu, Ahmet Uyar, Hasan Bulut, Geoffrey Fox

Community Grids Laboratory, Indiana University

wewu@indiana.edu, auyar@mailbox.syr.edu, hbulut@indiana.edu, gcf@indiana.edu

Tel: **812-8561245**

Indiana University Research Park, 501 North Morton Street, Suite 224, Bloomington, IN47404

Keywords: SIP, XGSP, Instant Message, Videoconference, Web-Services

Abstract

In order to build an integrated collaboration system over heterogeneous collaboration technologies, we proposed an colaboration Web-Service framework – XGSP and initially applied it to audio-video (A/V) conferencing.. SIP is a very important collaboration standard, which has been adopted by many large companies for IP telephoning, instant messaging as well as desktop videoconferencing. In this paper, we develop a XGSP based system to integrate SIP collaboration resources with the A/V subsystems we originally studied. This prototype system can support multiple clients and servers from different collaboration technologies such as SIP and AccessGrid as well as H.323 in the same conference.

1. Introduction

Collaboration and videoconferencing systems have become a very important application in the Internet. There are various solutions to such multimedia communication applications, among which H.323 [7], SIP [12], and Access Grid [1] are well-known. H.323 is an umbrella standard designed by ITU for multimedia conferencing over IP-based networks. It has been widely adopted by the industry of videoconferencing which produces many H.323 based systems. SIP is a standard of IETF, which is an alternative solution to H.323, especially for Voice over IP. SIP has been applied in IP telephoning, Instant Message (IM) and videoconferencing. Access Grid is a derivation from MMUSIC conference [4], which uses MBONE tools to support large scale videoconferences based on a high-speed multicast network.

It would bring substantial benefit to Internet users if we can build an integrated collaboration environment, which combines conferencing, streaming, instant message as well as other collaborations into a single easy-to-use, intuitive environment. However, at present all these systems have features that sometimes can be compared but often they make implicit architecture and implementation assumptions that hamper interoperability and functionality. Therefore it is very important to create a more general framework to cover the wide range of collaboration solutions and enable users from different communities to collaborate.

In the ref [2], we propose a Web-Services framework, named XGSP (XML based General Session Protocol) for an audio/video collaboration to solve the issue of heterogeneity and integration. Based on this framework, this paper focuses on how to integrate SIP collaboration resources, and how to make SIP clients or servers communicate with AccessGrid as well as H.323 systems.

The paper is organized in the following way: In Section 2, the XGSP framework and the design issues about SIP Web-Services are introduced. Section 3 presents the design of SIP Web-Services prototype system in detail. The implementation and some application examples are introduced in section 4. And we give the conclusion and future work in section 5.

2. XGSP & SIP Web-Services

2.1 SIP Architecture and Service

The Session Initiation Protocol (SIP) defines how to establish, maintain and terminate Internet sessions including multimedia conferences. The architecture of a SIP based system is showed in Figure 1.

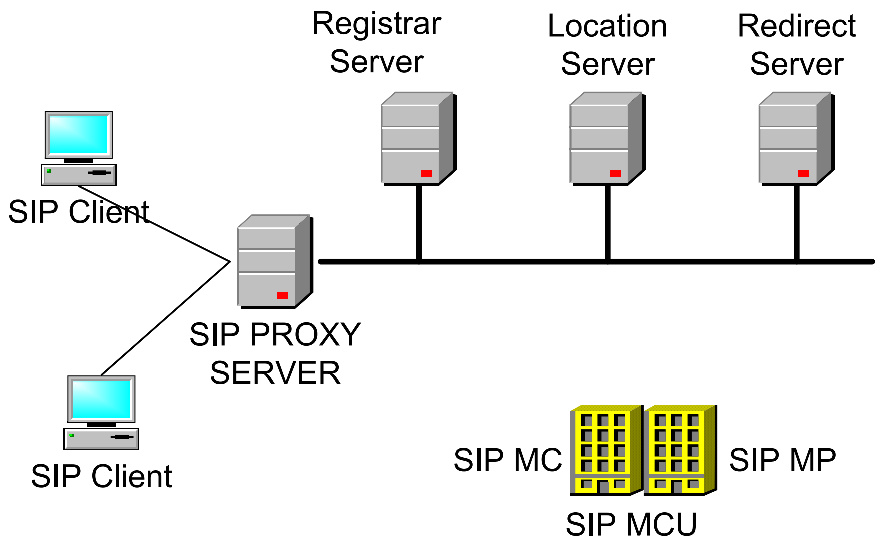

Fig 1 SIP System Architecture

A SIP system usually includes SIP client, SIP Proxy Server, Registrar Server, Location Server, Redirect Server as well as SIP Multipoint Control Unit (MCU). A SIP Proxy Server primarily plays the role of routing, enforcing policy of call admission. A proxy interprets and if necessary, rewrites specific parts of a request message before forwarding it. A SIP registrar accepts REGISTER requests and places the information it receives in those requests into the location service for the domain it handles. In addition, SIP Proxy provides instant messaging service, forwarding SIP Presence Event Message and SIP text Message to SIP clients.

2.2 XGSP Web-Services Framework

In this framework, we divide an A/V collaboration system into three parts: clients, session servers and multi-point communication channels provider. We need to connect the components from different system in two ways: firstly the system allows each client whether it is MBONE, H323 or SIP to build RTP channels with the multi-point communication RTP channel providers; secondly the system connects all the channel providers together to form a global RTP communication infrastructure.

The first goal requires a common session protocol, which enables different types of clients and servers to understand and communicate with each other. We define a XML based protocol, called XGSP (XML Based Session Protocol). Each native session command in SIP, H.323 as well as Access Grid is transformed into XGSP methods and executed by XGSP servers. Also the XGSP response is transformed back into the native response and returned to the native clients. The middleware architecture is required for the second goal to integrate various RTP multi-point communication servers. Web-services technology seems to be the best candidate for this purpose since it can run across various platforms and is easy to extend and understand.

In order to build a SIP web-services based on XGSP framework, we need to solve the following issues:

- The SIP calling procedure has to be translated into XGSP messages. INVITE, BYE as well as OK response to them must be transformed to Join/Leave Session Command in XGSP.
- SIP URIs has to be mapped into XGSP name space. Each SIP Client should make a registration in

the SIP Registrar of local domain. All of these registrations must be kept in the name storage of the XGSP system.

- SIP Instant Message services have to be integrated with Chat Room services of other collaboration environments. The SIP framework can support IM and presence services, usually implemented in the SIP Proxy. IM allows users to chat in a real-time way and the presence service notifies users of the status of their buddies. It is very useful to integrate them into XGSP collaboration systems. However Instant Message mostly works for private chat between two or more people. Only persons in the buddy list can join or be invited into IM session. So IM is suitable for informal and ad-hoc chat rather than room-based or scheduled chatting. XGSP framework mainly supports formal and scheduled collaboration, which requires a chat room environment hosting many participants in one or many discussions. Therefore, we must find a way to integrate both of the collaboration patterns.
- XGSP framework can not only integrate SIP clients but also other SIP communities into the whole collaboration system. SOAP connections should be established between the XGSP Manager Server and other SIP collaboration server so that SIP MCU as well as SIP proxy server can be commanded to connect with XGSP servers. Further since SOAP communication is slower than other protocols such as TCP, we have to ensure it will not affect the performance of the collaboration.

2.3 XGSP Servers for integrating various collaboration

We have designed a XGSP prototype system, in which there are two servers supporting the main framework of XGSP: Web Server and XGSP MCU. XGSP MCU is some kind of bridge between H.323/SIP endpoints and Access Grid clients, supporting both centralized channel and decentralized RTP channel in the same session. For those unicast-only H323/SIP endpoints, XGSP provides unicast audio and video channel, and perform video switch, video-mixing and audio mixing services. And XGSP MCU runs an agent in an IP multicast group to communicate with Access Grid clients.

The XGSP Web server provides a collaboration portal to users and the system administrator. Users need the web portal to participate in the audio/video collaboration. The user Web Portal supports the high-level services of XGSP collaboration framework, including XGSP registration, XGSP membership control as well as XGSP session control. System administrator can use the web portal to manage the system, for example configure the components of the system, and upload new services.

3. Design of SIP Web-Services

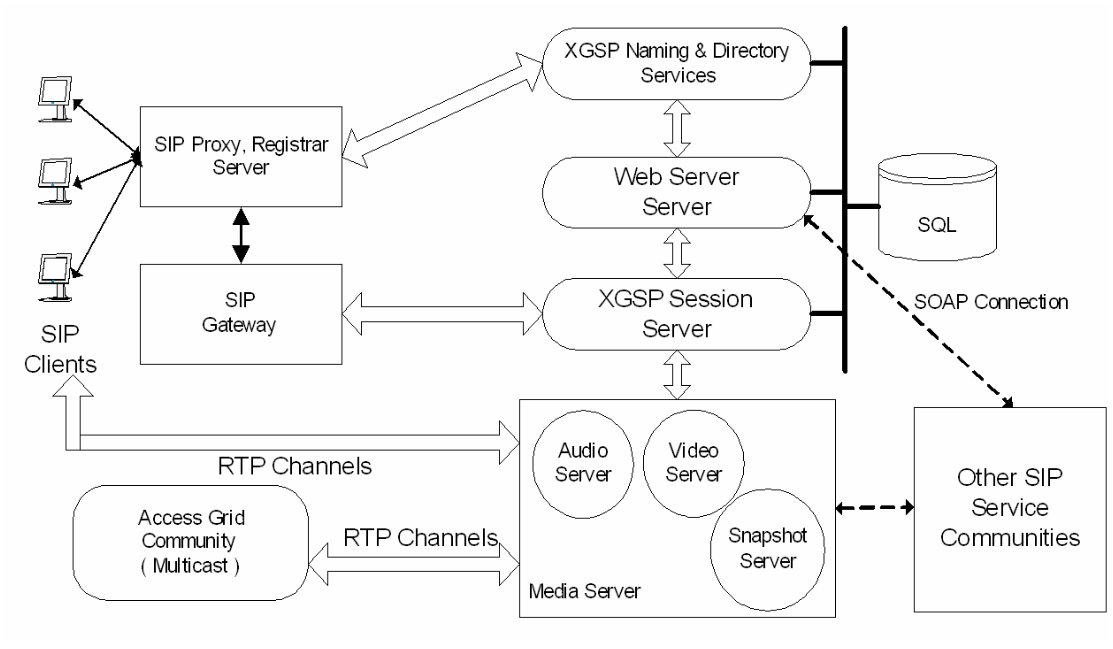

Fig 2 SIP Web Services Architecture

3.1 SIP-to-XGSP

The figure 2 shows the architecture of SIP Web-Services. We combine the SIP proxy and SIP registrar into a single server, and divide the XGSP into two different parts: XGSP Session Server and XGSP Media Server, which plays the roles of Multipoint Controller and Multipoint Processor. The XGSP Web Server can send commands to Session Server and Naming & Directory Server with the support of SQL database storage. The SIP Gateway works as a protocol bridge between SIP Proxy and XGSP Session Server, redirecting the RTP channels of SIP clients to XGSP Media Server.

The signaling translator maintains a dedicated TCP connection with XGSP-Session Server for each incoming SIP client, which is called an XGSP-connection. SIP Messages including INVITE and BYE, can be translated into XGSP messages and transferred over this connection. The procedure is as follows: when the SIP gateway accepts the INVITE request from a SIP client, it will send a JoinSession message to the XGSP Session Server, and get the media description by parsing SDP body in the INVITE message. Once the XGSP Session Server permits the connection, the SIP gateway will reply OK message with SDP of XGSP MCU to the SIP client. Further when the SIP gateway receives BYE message, it will send a LeaveSession message to Session Server and reply OK to the SIP client after it close XGSP connection.

According to the SIP standard, the registration creates bindings in a location service for a particular domain that associate an address-of-record URI with one or more contact addresses. Further the SIP registrar usually acts as the front end to the location service for a domain, reading and writing mappings based on the contents of REGISTER requests. We rely on XGSP naming & directory service to provide our location resolving, which maintains a database for the whole name space of the XGSP collaboration system. Therefore, all the registrations will be forwarded to the XGSP naming server for storage.

The SIP Gateway needs to register several records for different active collaboration sessions. For example, if there are three sessions are created in Web Server, the SIP Gateway has to register in SIP Registrar Server with three session names. When a SIP client wants to join an active session, it must send an invitation to the SIP URL of that session.

3.2 Instant Messaging & Chat Room Services

The IETF IMPP working group define a standard protocol "Model for Presence and Instant Messaging" [9], so that independently developed applications of instant messaging and presence can interoperate across the Internet. SIP provides an Event notification mechanism and MESSAGE method to support the model defined in RFC2778. The SIP Proxy is used to route SIP MESSAGE as well as other SIP methods among SIP clients. So we need to build a specific SIP Proxy for IM services, which can be combined with SIP Registrar into a SIP Server.

Chat Rooms can be implemented by building message reflection function inside a SIP Gateway. Since a SIP Gateway works like multiple SIP clients, we can regard these SIP clients as chat rooms. Once a SIP client user starts an IM session with a XGSP session of SIP Gateway, it joins in the chat session of this XGSP session. SIP Gateway accepts the MESSAGE sent by this SIP client, and forwarded it to other SIP clients that have been connected to this session. Further, the functions of the chat session can be enhanced by introducing shell commands, as for example, the command for listing all the participants.

3.3 HearMe SIP Web-Services

To integrate other SIP communities into the XGSP system, we need a high-level control level on which the XGSP collaboration Manager Server can co-ordinate with the control servers in other SIP communities. Further the general A/V media channels are also necessary for connecting all the RTP channels in these communities and individual clients. Such an approach can separate the control from the multimedia transmission so that slow SOAP connection will only increase the delay of building collaboration session and not impact multi-media channels. The architecture for the high-level control plane is showed in Fig 3. The XGSP Collaboration Manager Server uses SOAP connections to send collaboration control commands to Web servers in SIP, AG (AccessGrid) and H.323 communities. The XGSP MCU serves the purpose of bridging all the RTP channels from these communities. Dedicated RTP channels can be established between XGSP MCU and SIP/H.323 MCU as well as AccessGrid multicast group.

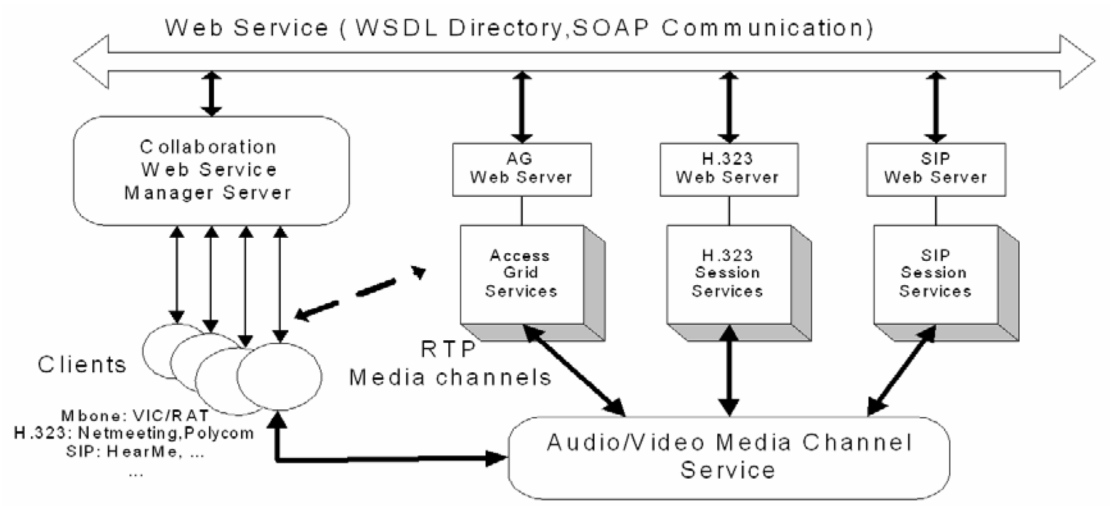

Fig 3 Integrating other SIP/H.323 Communities

A Web Server for the SIP community is supposed to provide capabilities for local session management and floor control. The collaboration manager server can invoke these APIs to create a collaboration session across different technology and administration communities, and control the collaborations.

For example, if a user wants to create a session named "testsession", for both AccessGrid users and the

SIP communities, he can define the XGSP session place as “AG: testroom; SIP: confroom” in order to connect two virtual conference rooms from the individual AG and SIP communities. When the meeting is about to start, the XGSP collaboration manager will activate this “testsession” by starting the AG agent in XGSP MCU to join in “AG:testroom” and invoking web services of SIP community so that the MCU in this community can dial in the XGSP SIP Gateway and connect to the XGSP MCU.

When a collaboration user logs and joins in XGSP session, he or she is required to choose which rooms they want to connect to and by which collaboration protocol. If they use VIC and RAT and want to enter AG: testroom, their VIC and RAT may be directed to the AG: testroom multicast session. If they use the SIP Messenger and want to enter SIP: confroom, their Messenger may be linked to the SIP MCU. Since the AG: testroom multicast room and SIP: confroom have been connected, all the clients in both of the virtual rooms can make audio collaboration. After the meeting is over, the XGSP collaboration manager can invoke web-services of the SIP community again to disconnect SIP MCU from XGSP MCU.

We use HearMe [6], a SIP based Voice-over-IP system, as an example to illustrate the SIP Web-Services interface. Similar interface can also be implemented based on other SIP or H.323 collaboration systems. The HearMe Talk Server plays the role of the session server in other systems and HearMe MCU provides SIP signaling and RTP channels for multipoint meeting. The HearMe Web services include: CreateSession, DestroySession, ModifySession, QuerySession, ConnectMCU and DisconnectMCU. All the commands are be defined in WSDL format and invoked by XGSP collaboration manager. Appendix A shows some segments of its WSDL definition, which describes the service of CreateSession.

4. Implementation and Examples

We have developed a prototype system to verify and refine our SIP web-services framework. It is developed in Java language and can be deployed in Windows or UNIX system. We follow JAIN SIP [8] stack specified by Sun for SIP development. NIST [10] is an open source project and implements SIP JAIN library. The NIST package also includes the examples of Proxy-Registrar Server and Instant Message client. We use the NIST library to develop SIP-XGSP Gateway and build our own Proxy Server based on the source codes.

The HearMe SDK provides an interface called Audio Service Control Protocol (ASCP) between HearMe Talk Server and HearMe MCU. ASCP is used to control the various aspects of the HearMe voice services. Based on ASCP, we build a HearMe Wrapper implementing HearMe Web Services interface by using ASCP to communicate with Talk Server. The AXIS SOAP processor [15] is used to process SOAP request and invoke the HearMe services through the HearMe Wrapper.

This prototype can support all the SIP compatible clients. We have tested HearMe client and Windows Messengers. Windows Messenger (MSN) is becoming very popular Instant Messaging client in the Internet, which can support text, audio, video collaborations and connect multiple kinds of servers, including standard SIP Proxy and Registrar Servers. We can configure Windows Messenger to make it registered in XGSP SIP Proxy and Registrar server, and attend the XGSP collaborations. The Fig 4 shows the scenario when two Window Messengers are talking with a RAT client and HearMe clients.

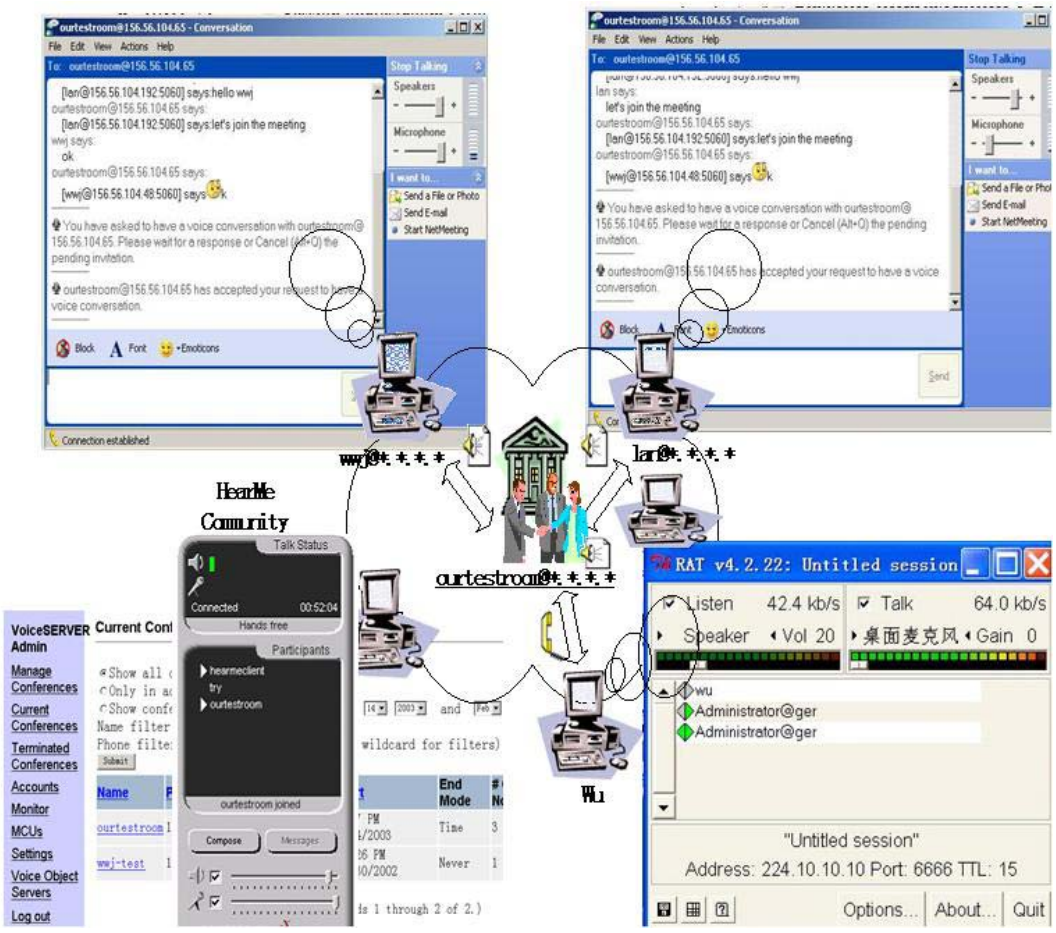

Fig 4 XGSP Chat&audio collaboration scenario

In this collaboration scenario, the system administrator has created a XGSP audio session, ourtestroom for AG:224.10.10.10/6666 and HearMe: ourtestroom. Both of the Messengers join the XGSP session "ourtestroom", and chat with each other in this session. And at the same time, they can also talk with RAT in the AG multicast session and HearMe clients in the HearMe conference.

5. Conclusion

In this paper, we present the design and implementation of a SIP Web-services system. Under the XGSP framework, this system can support SIP clients, Access Grid and H.323 clients in the same meeting, provide Instant Message and chat room services to Windows Messenger clients, and connect with other SIP communities. Such an integrated collaboration environment greatly benefits those users that want to enter the Access Grid world via SIP clients, merge the collaboration of videoconferencing and IM into a single environment, and create communication channels for different collaboration communities.

In our next development stage, we will enhance the capability of this system in the following aspects:

(1) In our prototype, a single XGSP MCU may become the performance bottleneck for very large-scale videoconference. NaradaBrokering [3], a distributed event middleware will be introduced to the system to implement a scalable and high-available A/V Event system [5].

(2) Our prototype supports room-based, scheduled collaboration while many peer-to-peer collaborations need Ad-hoc meeting mechanism. IM seems to be a very promising approach to realize Ad-hoc collaboration session. We will extend SIP IM to implement this new collaboration pattern.

6. Reference

[1] Access Grid, http://www.accessgrid.org
[2]Geoffrey Fox, Wenjun Wu, Ahmet Uyar, and Hasan Bulut A Web Services Framework for Collaboration and Audio/Videoconferencing, The 2002 International Multiconference in Computer Science and Computer Engineering, Internet Computing(IC'02), June 2002, Las Vegas
[3] Geoffrey C. Fox and Shrideep Pallickara, "The Narada Event Brokering System: Overview and Extensions", proceedings of the 2002 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'02)
[4] Handley, M., Crowcroft, J., Bormann, C. and J. Ott, "The Internet Multimedia Conferencing Architecture", Internet Draft, draft -ietf-mmusic -confarch-03.txt, July 2000.
[5] Hasan Bulut, Geoffrey Fox, Shrideep Pallickara,Ahmet Uyar and Wenjun Wu, *Integration of NaradaBrokering and Audio/Video Conferencing as a Web Service*, IASTED International Conference on Communications, Internet, and Information Technology, November 2002, US Virgin Islands.
[6] HearMe Audio conference system, http://www.hearme.com,
[7] H.323 ITU Recommendation
[8] Jain SIP, http://jcp.org/en/jsr/detail?id=125
[9] Model for Presence and Instant Messaging, RFC2778, http://www.ietf.org/rfc/rfc2778.txt
[10] NIST SIP, http://snad.ncsl.nist.gov/proj/iptel/.
[11] Real Time Transfer Protocol (RTP), RFC 1889, http://www.ietf.org/rfc/rfc1889.txt
[12] Session Initiation Protocol (SIP), RFC 2543, http://www.ietf.org/rfc/rfc2543.txt
[13] SIP-Sepcific Event Notification, RFC3265, http://www.ietf.org/rfc/rfc3265.txt
[14] Simple Object Access Protocol (SOAP) 1.1, http://www.w3.org/TR/SOAP/
[15] Steve Graham, Simeon Simeonov, etc, Building Web Services with Java, ISBN0-672-32181-5, Sams publishing.
[16] Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl

Appendix A

```
<?xml version="1.0"?>
<definitions name="HearMeSIPServices"
 targetNamespace="http://www.hearme.com/audiomcu/service"
 xmlns:hearme="http://www.hearme.com/audiomcu/service"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns = "http://schemas.xmlsoap.org/wsdl">
<!—Type definitions -- >
<types>
   <xsd:schema tagetNamespace="http://www.hearme.com/audiomcu/service"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      <xsd:comlexType name="scheduleDate">
        <xsd:sequence>
          <xsd:element name="StartSchedule" type="xsd:datetime" />
          <xsd:element name="EndSchedule" type="xsd:datetime" />
       </xsd:sequence>
      <xsd:complexType>
      </xsd:schema>
      ……
</types>
<!- - Message definitions -- >
<message name=”CreateSessionRequest”>
        <part name=”SessionId” type=”xsd:string” />
        <part name=”Max-Particpants” type=”xsd:string” />
        <part name=”schedule” type=”hearme: scheduleDate” />
        <part name=”AliasList” type=”hearme: AliasList” >
</message>
<message name=”CreateSessionResponse”>
       <part name=”result” type=”xsd:boolean”>
</message>
<! – Port type definitions -- >
<portType name=”HearMeSIPServicesPortType”>
  <operation name=”CreateSession”>
     <input message=”CreateSessionRequest”>
     <output message=”CreateSessionResponse”>
  </operation>
  <operation name=”DestroySession”>
  <operation name=”ModifySession”>
  <operation name=”QuerySession”>
  <operation name=”ConnectMCU”>
  <operation name=”DisconnectMCU”>
  …..
</portType>
<! – Binding definitions -- >
<binding name=”HearMeSipServiceSOAPBinding” type=”HearMeSIPServicesPortType”>
   <soap:binding stype=”rpc” transport=”http://schemas.xmlsoap.org/soap/http ”/>
   <operation name=”CreateSession”>
        <soap:operation soapAction=””/>
        <input>
               <soap:body use=”encoded” namespace=” http://www.hearme.com/audiomcu/service”
             encodingStyle=” http://schemas.xmlsoap.org/soap/encoding” />
        </input>
        <output>
               <soap:body use=”encoded” namespace=” http://www.hearme.com/audiomcu/service”
             encodingStyle=” http://schemas.xmlsoap.org/soap/encoding” />
        </output>
     … more binding definition ….
</binding>
<! – Service definition -- >
<service name=”HearMeSIPServices>
        <port name=”CreateSession” binding=”HearMeSIPServicesSOAPBinding”>
        <soap:address location=”http:// www.hearme.com:8080/axis/services/HearMeSIPServices” />
```

```
    </port>
</service>
</definitions>
```