Université d'Ottawa • University of Ottawa

Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Yanlin DONG

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

M. A. Sc. (Electrical Engineering)

GRADE - DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

Design and Implementation of SIB-based AD Hoc Conferencing System

A. Karmouch

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

B. Esfandiari

T. Yeap

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

SIGNATURE

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Design and Implementation of SIP-Based AD Hoc Conferencing System

By

Yanlin Dong, B. Eng.

A thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements of the degree of

Master of Applied Science in Electrical Engineering

Ottawa-Carleton Institute of Electrical and Computer Engineering
School of Information Technology & Engineering (SITE)
Faculty of Engineering

University of Ottawa
Ottawa Ontario, K1S 5B6, Canada

Your file Votre référence
ISBN: 0-494-01463-6
Our file Notre référence
ISBN: 0-494-01463-6

Abstract

The Session Initiation Protocol (SIP) is an IETF (Internet Engineering Task Force) standardized signaling protocol that can establish, modify, and terminate multimedia sessions or calls over the Internet. While incorporated with other protocols, such as Real-time Transport Protocol (RTP) and Session Description Protocol (SDP), SIP provides a wide range of Internet applications and services to users, including the multi-party conference.

Emerging technologies such as agent technology presents a new approach to empower users with advanced information processing methods and reduce the overload of users, particularly mobile wireless device users. A mobile agent is a type of agent that migrates from host to host under its own control to perform tasks, thereby adding flexibility to the system under consideration.

Combining SIP and agent technologies, a SIP-based ad-hoc meeting management system is proposed in the thesis. The SIP-based ad-hoc meeting management system aims to provide a dynamic real-time multimedia conferencing service to mobile users. In SIP-based ad-hoc meeting management system, a group of cooperating agents act on behalf of conference room entities and end-users, to dynamically create and maintain an ad hoc conference in the conferencing room under consideration. SIP helps to set up and maintain conference states. Using a precisely defined procedure, the system architecture and scenarios are designed to solve problems such as,

- How to start and manage an ad-hoc conference?
- How do users indicate their private constraints for private services?
- How do users initiate and manage a sidebar association in an on-going conference?
- How do conferencing participants identify other conference members and be notified of the changes made in an on-going conference?

The proposed SIP-based ad hoc meeting management system is capable of creating and managing conference sessions in an ad hoc environment, where participants are nomadic users. With several features, the SIP-based ad hoc meeting management system supports various services to users and provides mechanism to realize them.

Acknowledgments

I would like to express my gratitude to my supervisor, Professor Ahmed Karmouch for his support, advice and guidance. Without his great mentoring, I would not have completed this research successfully.

I would also like to thank Hamid Harroud for his advices. He provided me with the part of the Java implementation for SIP integration I needed in the thesis. His suggestion was greatly appreciated. Thank you to all of my colleagues at the Multimedia and Mobile Research Laboratory for their suggestions and cooperation. I extend my appreciation to NSERC and MITEL for their precious suggestions and monetary support that helped in completing my masters' program.

Finally, special thanks to my husband for his unconditional love and support through all my endeavors.

Contents

List of Figures

List of Abbreviation

ACL	Agent Communication Language
CA	ConferenceAgent
CPL	Calling Processing Language
CTA	Context Agent
FIPA-OS	Foundation of Intelligent Physical Agents – Open Source
GUI	Graphical User Interface
HTTP	Hyper Text Transport Protocol
PA	Policy Agent
RAT	Robust Audio Tool
RM	Room Manager
RTP	Real-time Transport Protocol
SAgent	SidebarAgent
SDP	Session Description Protocol
SIP	Session Initiation Protocol
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
VIC	Videoconference Tool
XML	Extensible Markup Language

Chapter 1 Introduction

1.1 Introduction

As the Internet and computer technologies grow towards their fully developed stage, the world is connected more closely than ever before and more communication activities rely on networks. Future networks will integrate traditional public telephone services with Internet services. As a result, all traditional telephone services will be provided via the Internet approach. The Internet Engineering Task Force (IETF) [1] has designed a signaling protocol namely the Session Initiation Protocol (SIP) [2] for Internet communications. SIP can establish, modify, and terminate multimedia sessions over the Internet including, Internet telephony call, multimedia audio and video conferencing, distance learning, and other multimedia applications. While incorporated with other protocols like Real-time Transport Protocol (RTP) [3] and Session Description Protocol (SDP) [4], SIP provides a wide range of real-time multimedia Internet applications and services.

Agent technology defines agents as autonomous software working on behalf of users. It has provided an alternative approach for software engineering to develop intelligent and autonomous systems. Mobile agents could migrate from one host to another host to perform tasks and provide personalized services for users in mobile environments.

Combing SIP and agent technology, a smarter and more powerful system can be formed. The thesis focuses on this 'new' view of designing a conferencing system. The goal of the thesis is to build a conferencing system in which SIP is used to manage an ad hoc conference and the overall conferencing activities are based on interoperations with agents in the conferencing room. Under consideration, each conference room entity in the conferencing system has the authority to control conference room activities and is fully in charge of the conferencing room. The room entity administrates not only the conferencing but also services in the conferencing room. Users automatically take part in an on-going conference when they enter the conferencing room, and conference participants are authorized to use services in the room depending upon their profiles. Room policies control the accesses to all the services. The users may bring their own services into the room and share those services with other conference participants. The respective owners govern the access of their 'brought in' private services. The room entity facilitates the communication between users and also among users and services.

The following part of this chapter discusses the motivation for designing the SIP-based ad-hoc meeting management system and describes the objectives and main contribution of this thesis. Then selected terminologies used throughout the thesis are also listed. Finally, the organization of the thesis is presented.

1.2 Motivation

Pocket sized small devices and notebook computers are proliferating with the development of computer technologies. This rapid growth coupled with ad-hoc

networking technologies have opened possibilities for mobile users to spontaneously participate in collaborative sessions while gathering at places, such as conference sites or meeting rooms. However, most research concentrates on the management of ad-hoc network, routing performance, predictability, etc. There is not much work done for conferencing management in the ad-hoc environments.

It has been a long time since people worked on conferencing applications providing services like real-time interaction and live information exchange to users. The earliest attempts date back to numerous tele-collaboration conferencing systems developed for packet-switched networks [5]. As the Internet continues to grow rapidly, various SIP-based conferencing models are proposed to adapt to the new communication approach i.e., multimedia communication over the Internet. Emerging technologies, such as agent technology, also are widely proposed for electrical commerce, home appliances control, and conference management.

Though SIP-based conferencing applications are already in existence, it still is a novel method to combine SIP and agent technologies into conferencing management. There are numerous advantages to encompass agent technology and IETF protocol SIP to design an ad-hoc conferencing management system, where participants act as mobile users with their laptops and/or other compact mobile wireless devices. The intelligence and autonomy of agents make the novel system more independent on environment changes and adaptive to dynamic ad-hoc networks. These properties provide mobile users more flexible and efficient services. The fact, that SIP is an Internet signaling protocol,

evidently leads the system under consideration towards the Internet adaptation. A conferencing management system combining SIP and agent technology could offer a smarter and more powerful system.

1.3 Objectives

The thesis proposes an agent-based ad-hoc conferencing system that uses SIP for conferencing management. The main objective of the thesis is to design a SIP-based ad hoc conferencing system and define scenarios for conferencing activities in the conferencing room under consideration. The objectives of the thesis are summarized as follows:

- Examine the list of requirements for the conferencing system and fulfill them.
- Propose an approach to effectively present an ideal conferencing system.
- Apply the proposed approach to a prototype and validate it.

1.4 Contribution

The main contributions of the thesis are as follows.

(i) *Conception and implementation of a SIP-based ad hoc conferencing system model.*

The thesis proposes a design that uses SIP and software agents to manage the ad-hoc conferencing. The proposed method assumes that all participating devices either support or are capable to support agents in the architecture.

(ii) Definition and realization of features of conferencing management system in one ad hoc environment.

A novel way is described for handling or managing an ad-hoc conference in the thesis, which satisfies the followings.

- It can start and manage an ad-hoc conference in the conferencing room under consideration;
- While enterprise policies exist for facilitating service accesses in the conferencing room, participants (with 'private' services) indicate their constraints for their private services;
- Participants can initiate and manage sidebar associations in an on-going conference;
- Conference participants can identify other conference members and can be notified about changes in an on-going conference.

1.5 Terminology Related to Thesis

This section provides an overview of selected important terms frequently used in the thesis.

- **Agent:** A software program that performs some set of tasks on behalf of a user or another program. It has one or more of the following characteristics: autonomy, pro-activeness, sociability, mobility, etc.

- **Mobile agent:** A program that acts on behalf of a user or another program and is able to migrate from one host to another host through a network to perform tasks [6].
- **Wrapper Agent:** An agent that dynamically connects to software/applications to invoke and fully control the operations on the software systems [7].
- **Agent Communication Language (ACL):** A common language among agents to 'talk' with each other and share information and knowledge so that an agent can share its actions or results with other agents [8].
- **Policy:** A set of conditions applied to agents or systems to trigger or prohibit certain actions.
- **Multimedia:** A general term indicating the operation and process of media types such as video, audio, text, animation, and graphics.
- **Conference:** A meeting among a group of people who have an agreed topic.
- **Ad-hoc Conference:** An improvised or impromptu conference in which people dynamically join for a specific purpose or to discuss a specific topic for a certain time and then dynamically leave.
- **Multimedia Conference:** A conference providing an electronic environment for people exchanging information and communicating by means of media types, such as audio and video.
- **Sidebar:** A sidebar is a "conference within a conference". "It is a conversation amongst a subset of the participants to which the remaining participants are not privy" [9].

- **Session:** A meeting devoted to a particular activity or for executing a groups' goal.
- **Multimedia Session:** A session which is "responsible for keeping track of various data streams associated with one multimedia communication (e.g. data, voice, video streams)" [10].
- **RTP media stream:** Real-Time Transport Protocol is a protocol designed by IETF to support real-time data transport such as video and audio streams. RTP media stream refers to media streams transported by using RTP [3].
- **RTCP:** Real-Time Control Protocol is an IETF designed protocol for conveying feedback and membership information in a RTP session [3].
- **SDP:** Session Description Protocol is designed by IETF. An SDP session description conveys information such as creator details, media titles, media types and transport addresses, etc [4].

1.6 Thesis Organization

The architecture of the SIP-based ad hoc conferencing system that enables users to create and manage conference sessions in an ad hoc environment is proposed with different features and mechanisms for realizing them.

The remainder of the thesis is organized as follows. Chapter 2 makes the thesis self-contained by introducing background knowledge of SIP and Call Processing Language (CPL) [11][12]. Firstly, it outlines the basic operations of SIP and addresses important SIP concepts involved in the conferencing system. Addressed SIP concepts

include Back-to-Back User Agent (B2BUA) [9], present service, etc. Afterwards, it briefly presents the Call Processing Language (CPL) and discusses some examples to show how CPL is used in describing telephony features. Chapter 3 describes virtual conference, multimedia conference, and characteristics of ad-hoc meeting. Furthermore, this chapter states conferencing models based on SIP, which are the basis of our SIP-based ad-hoc conferencing model. In chapter 4, the SIP-based ad-hoc conferencing systems' design is discussed in detail. The entities involved in the system are introduced first, followed by the description of scenarios in the set up of the conferencing system and other system features. The mechanism of starting a conference, defining users' source policies, creating sidebar sessions, and notifying changes in a conference to conference participants are fully discussed. Chapter 5 gives a brief description of the prototype implementation. The implementation details and results are discussed before the snapshots of the SIP-based ad hoc conferencing system. Finally, chapter 6 summarizes the thesis work and suggests directions for possible future research.

Chapter 2 Background

This chapter provides background information related to the thesis. As Session Initiation Protocol (SIP) [2] and Calling Processing language (CPL) [11, 12] are extensively involved in the design of SIP-based ad-hoc meeting management system, they are discussed in detail in this chapter.

Firstly, an overview of SIP is provided. The architecture, its major components, operation modes, and extensions of SIP are introduced in detail. This is followed by a brief description of CPL script language. The CPL scripting languages' structure and operation are browsed, and examples stating how CPL is used to describe telephony services are also provided.

2.1 Overview of SIP

The Session Initiation Protocol (SIP) is an application-layer controlling protocol that can establish, modify, and terminate multimedia sessions or calls. It has been standardized within IETF as a signaling protocol for establishing real-time calls and conferences over the Internet. It may be utilized in applications such as multimedia conferences, distance learning, and Internet telephony. SIP is similar in both syntax and semantics with Hypertext Transport Protocol (HTTP) [13]. It is a text-based protocol and can be easily extended. Since SIP is a general-purpose protocol, it is independent of packet layers and

supports both UDP and TCP. SIP incorporates with other protocols for multimedia communication and control. For example, it incorporates with Session Description Protocol (SDP) [4] for multimedia session description during SIP session establishment, and with real-time transport protocol (RTP) [3] for real-time data transportation after SIP session establishment.

2.1.1 Entities and SIP Address

The main entities in SIP are the User Agent, SIP Proxy Server, SIP Redirect Server and Registrar.

The SIP User Agent (UA) also known as SIP endpoint, functions as client, namely UAC (User Agent Client), when initiates requests and functions as server, namely UAS (User Agent Server), when responds to requests. UA communicates with other UA directly or via the SIP Proxy or Redirect Server. The SIP Proxy and Redirect Server provide different working modes in a session establishment. When a SIP UA request is received, the SIP Proxy forwards this request to the next SIP server or another UA within the network. A SIP Redirector however returns the requested server' address to the UA for the UA contacting the destination directly. A SIP Registrar is responsible for maintaining location service, namely storage of SIP UA addresses. The proxy and redirect servers check the destination SIP UA's address from the location service to help establish session between a caller and a callee. The SIP Proxy, Redirect Servers and Registrar are logical entities which may physically co-exist in one machine.

Every SIP entity has a unique SIP address to identify it. SIP address is presented in the form of SIP URL:

- sip: username @ domain

The user part is either a user name or a telephone number. The domain part is either a domain name or a numeric network address. While the user part is optional, the domain part is mandatory. The domain part may include parameters like user name, port number, etc. Some examples of SIP addresses are,

- yli@uottawa.ca
- +1-212-555-1234:2345@room.com; user = phone
- alice@10.1.2.3

SIP address is a globally reachable address. SIP users are bound to these addresses by frequently registering their latest locations or addresses to the SIP Registrar server. The location service will be looked up whenever a caller wants to establish real-time communication with a callee.

Besides using SIP URL, SIP users may also use e-mail addresses or telephone numbers as their additional aliases. SIP UA's can be identified or located by their aliases as well.

2.1.2 Methods

There are six request methods defined in SIP. They are INVITE, BYE, OPTIONS, ACK, CANCEL and REGISTER. The functions of these six SIP methods are summarized as follows,

- INVITE: SIP UA uses an INVITE request to initiate a session
- BYE: used to terminate a session between two users
- ACK: used to confirm establishment of a session
- CANCEL: used to terminate a processing call
- REGISTER: used to register a user's SIP address to the SIP Registrar server
- OPTIONS: used to query server capabilities without really setting up a call

The most important method is the INVITE. A SIP user initiate a call by issuing an INVITE request. When a caller wants to have a session with a callee, his/her SIP UA sends out the SIP INVITE to the particular callees' SIP UA. If the callee accepts the call, his/her SIP UA will give back a "200 OK" response to the caller. After getting "200 OK" response, the caller will then generate an ACK request to the callee to confirm session establishment. This finishes the process of session establishment and users can now use other protocols like RTP for real-time media communication. A new INVITE request will create a new session. A SIP user may change an existing session by re-issuing a new INVITE.

The REGISTER request is used to update a user' current contact address. By sending a REGISTER request to the SIP Registrar, A SIP UA may update its latest reachable location and address in the SIP server.

2.1.3 Message

In SIP, requests and responses between clients and servers are called SIP messages, which include two parts, namely, message header and message body. Each SIP request consists of a set of header fields that describe the call in general. The message header gives a full description of a request including request methods, callers' address, and path taken by the request so far. The most important header fields are TO and FROM, which contain callee and caller address respectively.

The message body follows the message header, which describes the individual media session of a call. SIP requests may optionally contain message bodies, which either have session descriptions or other service descriptions. SIP uses Session Description Protocol (SDP) for media description, which conveys information in textual format. In the media description, users can negotiate *media type* and *encoding* used for the communication. SIP has no restrictions to any particular media type. A multimedia session can contain any media streams, such as audio, video, text, etc. When a call is set up using SIP, the SIP INVITE message contains SDP session description defining media format used. The media destination address is also indicated in SDP.

An example of SIP INVITE message with SDP media description is given as follows,

Figure 2.1 SIP message example

2.1.4 Architecture

The basic architecture of SIP is client/server in nature. As mentioned earlier, SIP UA works at client end and updates users' contact information to the SIP REGISTRAR frequently. For SIP UAs creating calls with each other, SIP may operate in two modes namely, proxy mode and redirect mode.

2.1.4.1 Registration

In SIP, the Registrar is responsible for maintaining registration information of clients. UA's send registration information containing their current network locations to local SIP

Registrar, which in turn stores received information in a location service. Whenever 'looked up', the Registrar will retrieve appropriate information and send them back to appropriate UA or SIP server.

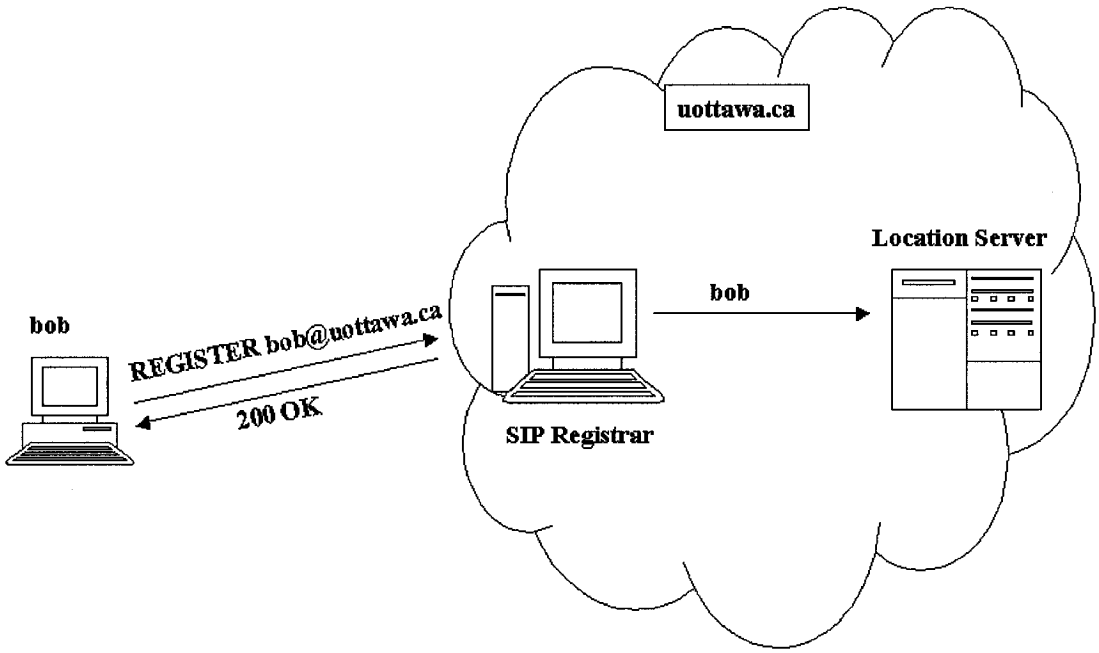


Figure 2.2 SIP Registration

As seen above in figure 2.2, SIP user 'Bob' sends a SIP REGISTER request to a SIP Registrar indicating his current contact address. The SIP Registrar stores received address in location server. After registration is successful, the Registrar gives "200 OK" back to 'Bob'. When there is a new call to 'Bob', his contact address will be 'looked up' from the location server and all calls will be forwarded to 'Bob's contact address.

2.1.4.2 Operation in Proxy Mode

When a SIP Proxy receives requests from clients, it forwards requests to the next SIP server or destination UA (if available within the network). SIP Proxy servers can either be stateful or stateless. When SIP servers are stateful, they will remember all states about a request. Stateless servers forget everything about a request after they forward the request.

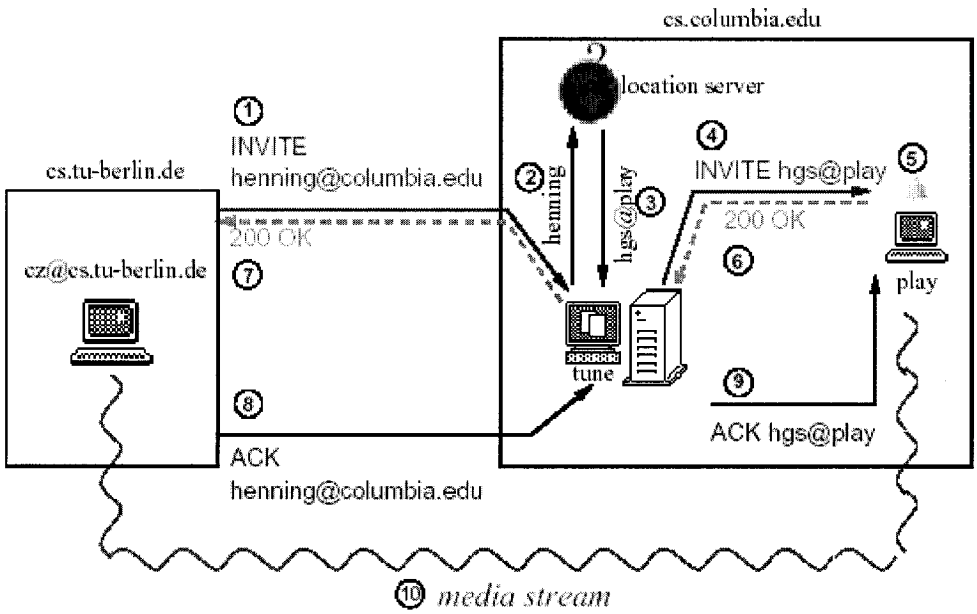


Figure 2.3 SIP operation in proxy mode [14]

Above shown figure 2.3 represents an example of SIP working in proxy mode. A user at cz@cs.tu-berlin.de sends a SIP INVITE request to henning@columbia.edu. The request is first sent to proxy "tune" (1). By looking up the address list in location server, the proxy determines that the callee is currently reachable at address hgs@play (2)(3). So

the proxy forwards the caller's INVITE to hgs@play (4). After acknowledge signal is received, the session is set up between users at cz@cs.tu-berlin.de and hgs@play.

2.1.4.3 Operation in Redirect Mode

When there is an incoming request from clients, the SIP Redirect Server responds differently with a SIP Proxy. At first, the Redirect Server also looks up the location server for destination address. Then, instead of forwarding the request directly to the destination address, the Redirect Server gives requesting clients the requested server address or the destination address, so that the requesting client may contact the destination directly.

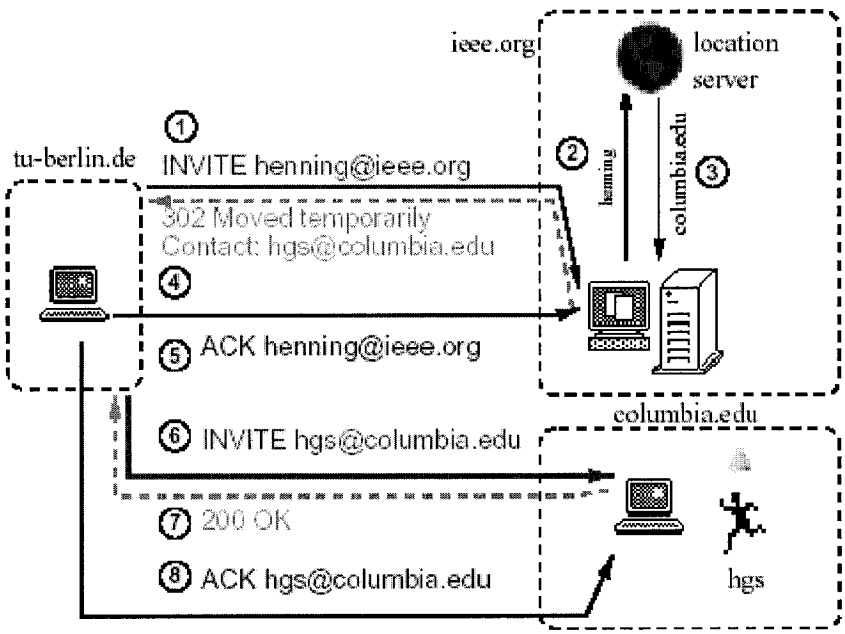


Figure 2.4 SIP operation in redirect mode [14]

In the above shown figure 2.4, user at cz@cs.tu-berlin.de tries to establish communication with user at henning@ieee.org. First, he/she sends his/her INVITE request to a Redirect server (1). By looking up the address list in the location server, the Redirector server gets callee's current contact address hgs@columbia.edu (2)(3). Then the Redirector server gives callee's contact address back to caller (4). The caller sends another INVITE request directly to the callee using the address received from the Redirect server (6). After acknowledgement is received (8), the session between user cz@cs.tu-berlin.de and user henning@ieee.org is established.

2.1.4.4 Transaction

When SIP request are created by UACs, there are responses generated by SIP servers and UASs following the requests. A request and its following responses are together called a transaction. A typical SIP transaction is showed below figure 2.5.

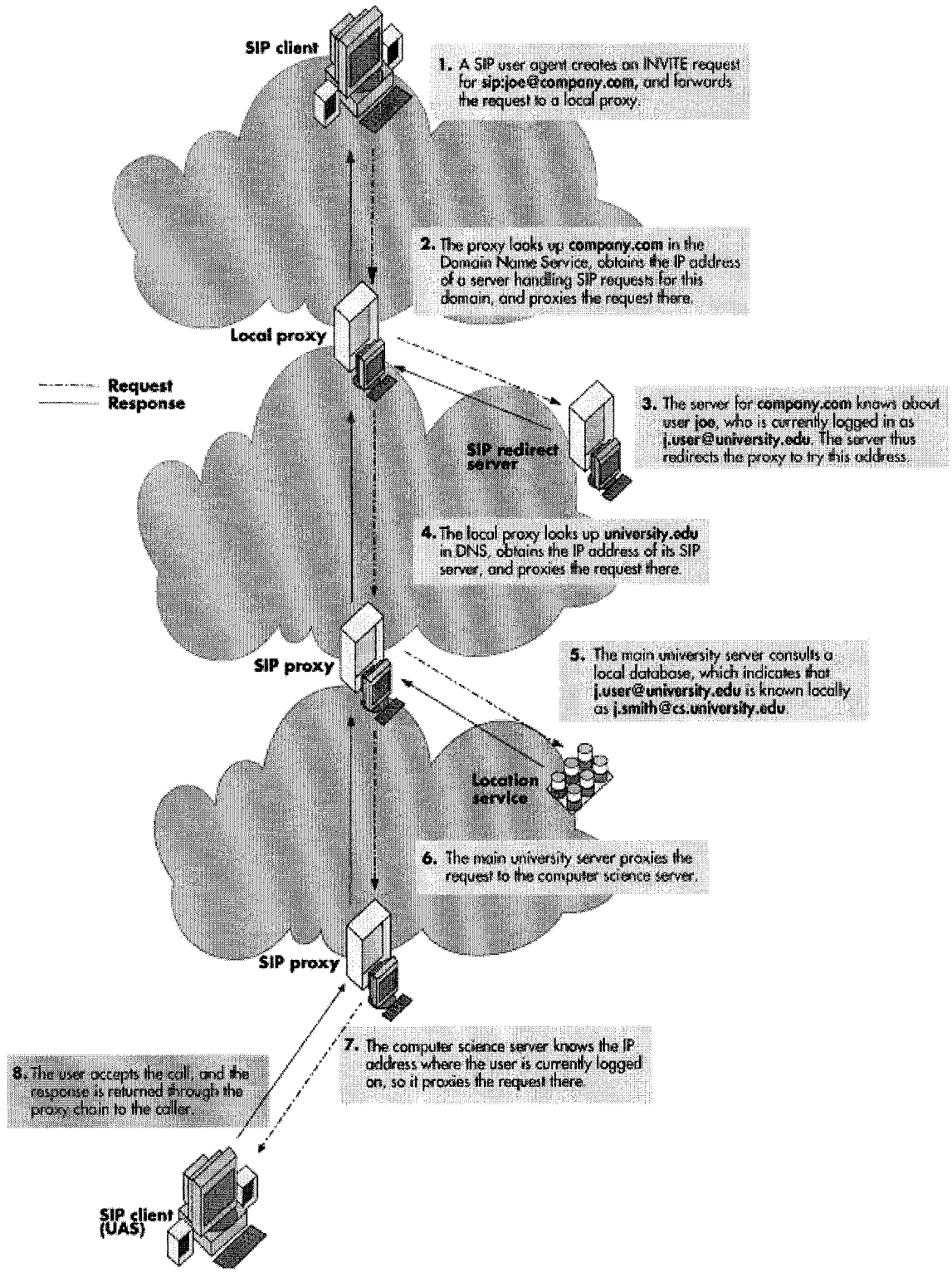


Figure 2 5 A typical SIP transaction [15]

2.1.5 Extensions

One main advantage of SIP is its extensibility. SIP can easily be extended to provide various services. It has extensions for services like presence services [16], instant messaging [17], call transfer [18], call control [19], etc.

2.1.5.1 Back-to-Back User Agent (B2BUA)

In SIP, a special entity called Back-to-Back User Agent (B2BUA) [20] works differently with a traditional UA or proxy. The B2BUA can be viewed as virtual UAS/UAC connected back to back. It acts as User Agent Server (UAS) when it receives a SIP request. Then, unlike normal SIP UA it processes the received request according to some internal functions and determines how the request should be answered. It then acts as User Agent Client (UAC) and generates a new request corresponding to result obtained and finally sends it out. The mechanism is shown below in figure 2.6.

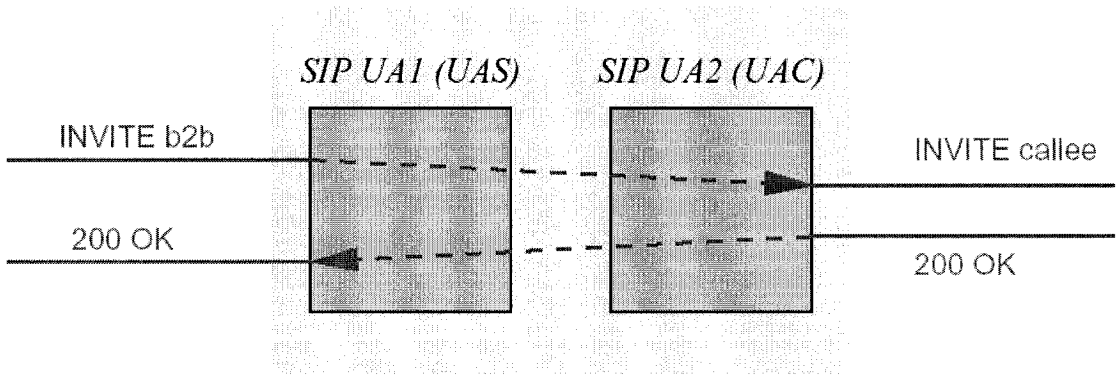


Figure 2.6 B2BUA [14]

2.1.5.2 Presence Service

In the SIP presence service, the entity 'Presence Agent' (PA) is capable of handling incoming SUBSCRIBE requests for a presentity [16]. Another entity 'Presence User Agent' (PUA) is responsible for a user's presence information namely, presentity information. When the presentity information changes, PUA should inform the PA regarding the changes. In the case of the PA being collocated with a SIP server, PUA sends presentity information to the SIP server by SIP REGISTER to notify the PA. In another case, the PA is co-located with the PUA, so the PA knows presentity information by default. After receiving presentity information from the PUA, PA will generate NOTIFYs to all the subscribers.

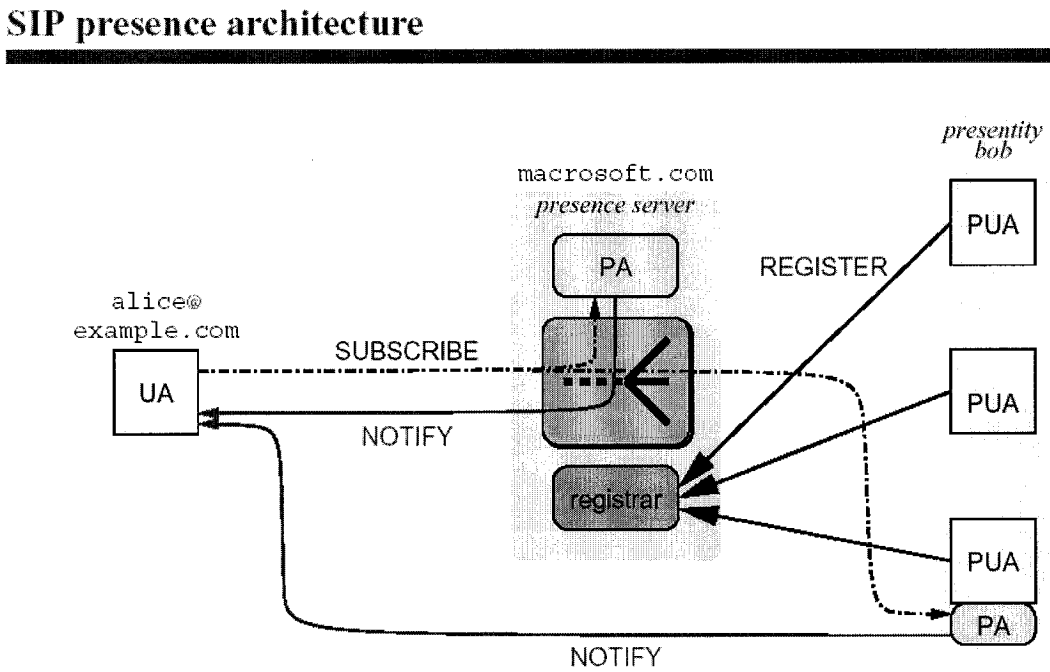


Figure 2.7 SIP presence service [14]

2.2 Overview of CPL

The Call Processing Language (CPL) [11, 12] is a script language defined by IETF. It is based on XML and is used for describing and controlling Internet telephony services. CPL is independent of any signaling architecture or protocol. It provides an approach for un-trusted users to define their own services in the network.

CPL script should be defined by end users and then uploaded to network servers. The logic will be verified and executed in the server. It is the server's responsibility to validate the script and ensure the safety of the execution.

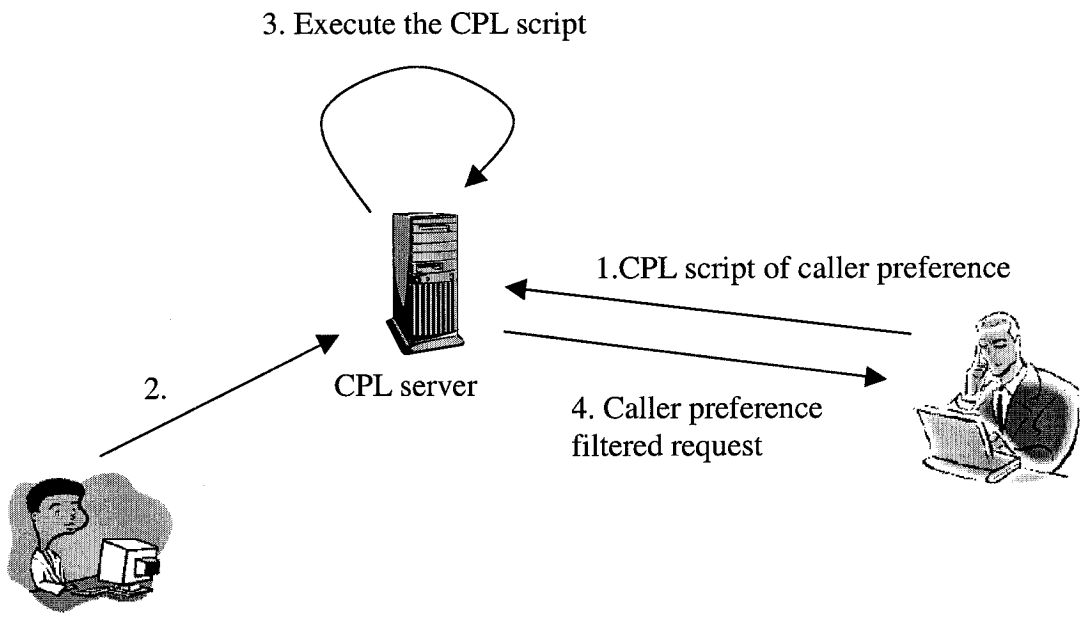


Figure 2.8 CPL operation

For example, in above shown figure 2.8, the user defines caller preference in CPL script and uploads the script to the CPL server (1). Server validates the script when it is

received. Later, there is an incoming call to this user (2), the server first checks whether the user has any caller preference script. If 'yes', the server will retrieve the script and executes it (3). All incoming calls will be filtered as defined in the CPL script. Only calls satisfying the users' preference are forwarded to the corresponding user (4).

The main advantage of CPL is that it is powerful for users to describe a larger number of services and features; however at the same time, it limits users' capability to do anything more advanced.

2.2.1 CPL Script Structure

A CPL script describes certain call processing actions and can be represented as a structured tree as shown below in figure 2.9. As seen in figure 2.9, the tree contains boxes representing nodes and lines with arrows representing outputs. The nodes and outputs are described by the XML tags. While nodes specify decisions and behaviors, outputs specify action results of nodes and lead to further actions and decisions. The structure tree describes server's operations and decisions during call set-up for both incoming and outgoing calls.

There are four CPL nodes called switch nodes, location nodes, signaling operations and non-signaling operations. Among them, ***switch nodes*** consist of a list of conditions and show different choices that a CPL script can make. Different decisions lead to different outputs, thereby leading to different nodes, which in turn will result in

different actions being executed in forthcoming steps. For example, in call forwarding, a script may be defined in such a way that incoming calls from user A will be denied automatically, but if a call is coming from 'boss', it will be automatically forwarded to owner's cell phone.

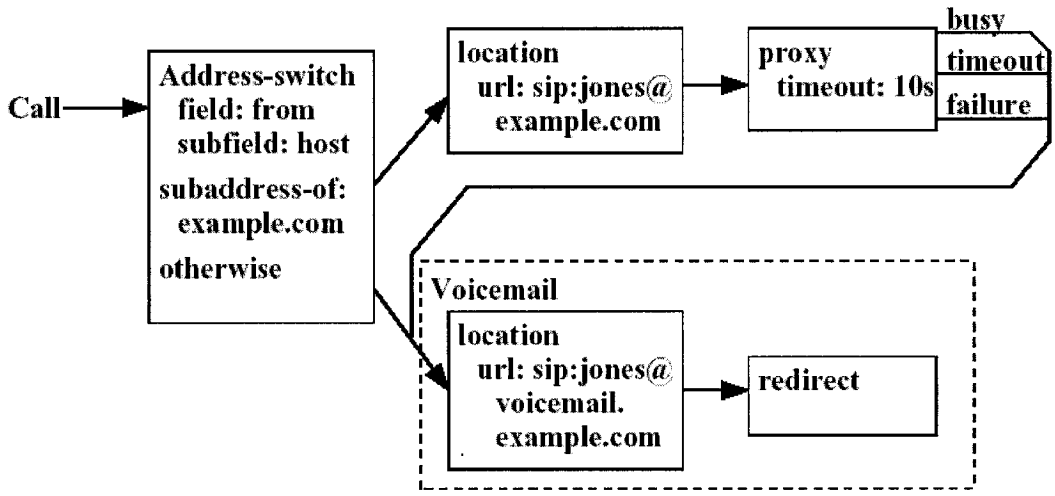


Figure 2.9 Sample CPL Action: Graphical Version [12]

The ***location nodes*** specify locations that should be contacted for subsequent actions. This 'specified and contacted' location may be given an explicit address in the form of URL, or the server gets it from indirect lookup from source, such as a database server or the SIP Registrar server. A CPL script has an implicit location set, and location nodes may add locations or remove locations from the implicit location set.

Signaling operations nodes define the behavior for three signaling operations namely, proxy, redirect and reject. "Proxy" triggers the CPL server to forward the call to currently specified location and wait for a response. If the call is successful, execution of

CPL script is terminated, otherwise, node outputs like "busy", "noanswer" or "failure" will be executed. "Redirect" causes the CPL server to redirect the request to currently specified location and the script is thus done. "Reject" indicates the server to reject the call by optionally giving a reason.

Non-signaling operations allow the server to act independently in regard to signaling protocol. For example, the CPL server may record the calling information in a log so that users can check them later; or the CPL server sends e-mails to users notifying event occurrence.

2.2.2 Examples

Two CPL examples are presented in this section to show how CPL is used to express user control information.

As seen below in figure 2.10, the "subaction" tag has a location node defining an URL for user Jones' voicemail. Following the "incoming" tag, top-level actions are defined for incoming calls. First, the address of the script owner is specified by another location node. When a call is received, the "proxy" tag indicates the server to contact Jone until 'timeout' parameter (8 seconds) is reached. The operations for all possible exceptions like 'Jones is busy', 'no answer', or 'time out' are indicated to the server as well. Finally, the server forwards the call to Jone's voicemail as defined in "subaction".

```
<?xml version="1.0" ?>
  <!DOCTYPE cpl PUBLIC "-//IETF//DTD RFCxxxx CPL 1.0//EN" "cpl.dtd">

  <cpl>
    <subaction id="voicemail">
      <location url="sip:jones@voicemail.example.com" >
        <proxy />
      </location>
    </subaction>

    <incoming>
      <location url="sip:jones@jonespc.example.com">
         <proxy timeout="8">
           <busy>
             <sub ref="voicemail" />
           </busy>
           <noanswer>
             <sub ref="voicemail" />
           </noanswer>
         </proxy>
      </location>
    </incoming>
  </cpl>
```

Figure 2.10 Example Script: Call Forward Busy/No Answer [12]

The example shown below in figure 2.11 is more complex in comparison with figure 2.10. If user Jones is busy, his calls are forwarded to his voicemail. If he doesn't answer phone calls within certain time, calls from his boss are led to his cellular phone "19175551212", and all the other calls are forwarded to his voicemail.

```
<?xml version="1.0" ?>
<!DOCTYPE cpl PUBLIC "-//IETF//DTD RFCxxxx CPL 1.0//EN" "cpl.dtd">

<cpl>
  <subaction id="voicemail">
    <location url="sip:jones@voicemail.example.com">
      <redirect />
    </location>
  </subaction>

  <incoming>
    <location url="sip:jones@phone.example.com">
      <proxy timeout="8">
        <busy>
          <sub ref="voicemail" />
        </busy>
        <noanswer>
          <address-switch field="origin">
            <address is="sip:boss@example.com">
              <location url="tel:+19175551212">
                <proxy />
              </location>
            </address>
            <otherwise>
              <sub ref="voicemail" />
            </otherwise>
          </address-switch>
        </noanswer>
      </proxy>
    </location>
  </incoming>
</cpl>
```

Figure 2.11 Example CPL Script: A Complex Example [12]

2.3 Summary

This chapter provided background knowledge related to the thesis. First, the protocol SIP is discussed in detail. This is followed by CPL descriptions. From the topics discussed in this chapter, it can be concluded that SIP and CPL provide fundamental tools and techniques for designing conferencing system presented in the thesis.

The next chapter discusses conferencing applications in detail. It first introduces multimedia conference followed by media transportation and ad-hoc meetings. SIP applications and various SIP-based conferencing models are also elaborated. The chapter discusses why SIP should be used for the ad-hoc conferencing management system presented in the thesis. Finally, a general description of ad-hoc conferencing management system is presented.

Chapter 3 SIP and Conference

This chapter explores issues regarding conferencing. First, a brief overview of virtual conference and multimedia conference will be addressed and their definition and characteristics are discussed. Then a brief introduction is given about media transmission in conferencing and protocols used for media transportation. This chapter further introduces ad-hoc conferencing and issues related with management of ad-hoc conferences. This chapter gives significance to SIP applications and examples of SIP applications will also be discussed. Emphasis will be made on SIP-based multi party conferencing models, wherein SIP-based ad-hoc conferencing system presented in the thesis stems from. The SIP-based ad-hoc meeting management model will be briefly described. Why SIP is chosen for conferencing management and what are its main differences while compared with existed models are presented as well.

3.1 Virtual Conference

According to E. Shapiro [21], a Virtual Place is "a foundation for Internet-based communication and collaboration". The Internet provides "a plethora of applications for human interaction", it offers a virtual place for humans to communicate and interact with each other.

Unlike in a traditional conference wherein people should be physically present, virtual conference participants are physically separated along networks, but virtually join a meeting where they can have "live" interaction and information exchange.

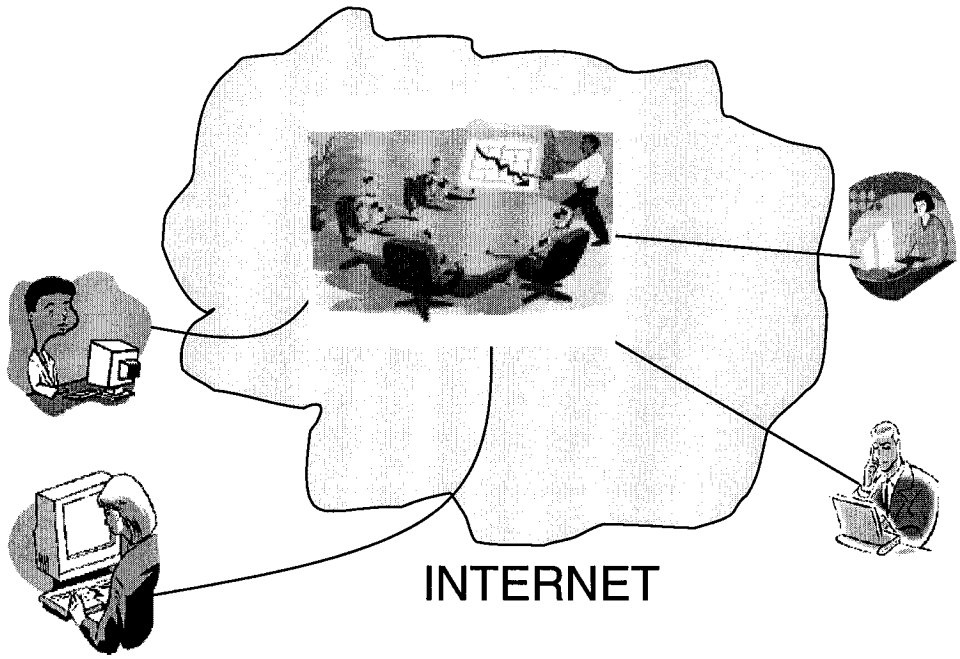


Figure 3.1 Virtual Conference

Virtual conference applications include audio and video conference, common multimedia conference; and ad-hoc conference. An ad-hoc conference is a dynamic meeting based on ad-hoc networks where users randomly join or leave the network.

3.1.1 Multimedia Conference

The word 'multimedia' in multimedia conference refers to computer-based media such as text, animation, graphics, and real-time audio and video. According to E. M. Schooler

[5], "multimedia conferencing in the broadest sense is using of mixed media for group collaboration". Since a multimedia conference is one type of virtual conference, it has the characteristics of virtual conference i.e., participants co-operate even though they are remote to each other. Multimedia conferencing provides people a meeting service for real-time communication and information exchange by means of media like audio and video. It is a synchronous conference that is "intended for simultaneous users who have real-time interactions".

Efforts to design multimedia conferencing applications date back to teleconferencing. There are numerous telecollaboration systems that have been developed for **packet-switched and circuit-switched networks** with multimedia control [5]. Noteworthy system includes BBN's MMConf system [22, 23], a distributed real-time conferencing system which integrates variety of applications such as, multimedia editor, presentation tool, etc. MMConf system is used among remote locations and concentrated to accommodate remote conferencing with inter-office or inter-meeting-room nature. Bellcore's Cruiser project [24] specified an unplanned informal real-time audio and video teleconferencing system, which targeted on solving several disadvantages posed by physical proximity. For example, distances like "around the corner", "over one hallway" are enough to hinder communication between users. Casner's Multimedia Conferencing project [25, 26] focused on real-time and multi-site conferences. Different from other projects, Casner's system "targeted remote conferencing across transcontinental packet-switched networks".

As the Internet continues to 'boom', users are becoming more interested in alternative infrastructures to support multimedia services including multimedia conferencing. For the Internet conferencing and telephony, the International Telecommunication Union (ITU) recommended H.323 [27, 28] as the signaling control protocol. However, it only supports TCP and also is too complex as the specifications of H.323 have more than 300 pages. IETF designed a signaling protocol namely SIP (Session Initiation Protocol) for initiating and controlling multimedia sessions over the Internet. As discussed before, SIP supports TCP, UCP, and also multi-point communication. The main advantages of SIP include its simple specifications, service extension support, and flexibility to adapt integrations.

3.1.2 Real-Time Media Transmission

Users can exchange information in a multimedia conference including media streaming such as audio and video streams. They could be transmitted simultaneously for communication. A live real-time multimedia communication requires new Internet services for real-time data transport and control. IETF designed protocols for the purpose of managing delivery and 'current' multimedia data over the Internet. RTP [3] (Real-time Transport Protocol) provides services for real-time data transport such as video and audio. RTCP (Real-Time Control Protocol) is a standard protocol for monitoring real-time data delivery quality. Both RTP and RTCP are independent of transport and network layers. These two protocols interoperate to guarantee smooth data transmission over the Internet.

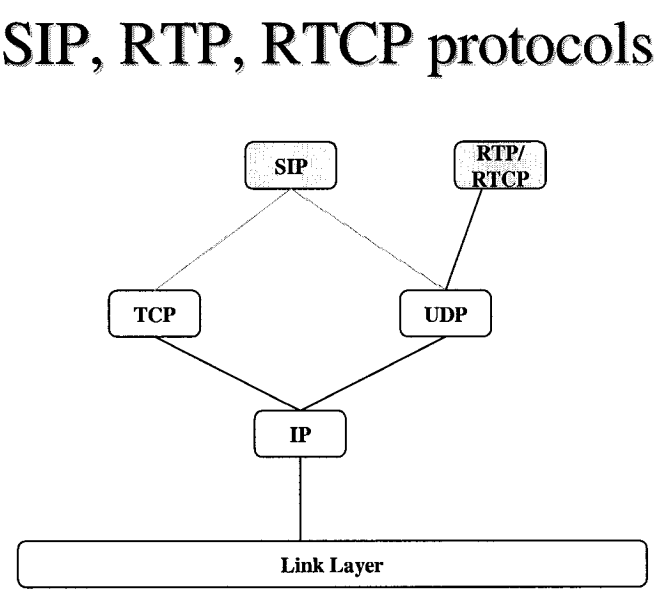


Figure 3.2 SIP, RTP, RTCP protocol

RTP supports both multicasting and unicasting, but it is primarily designed for multicast real-time data transmission. Media transmissions involving media streams often called "RTP stream" are controlled by RTP. RTP packet header includes time-stamping, sequence numbering, and other octets to take care of timing issues. It provides services including "time reconstruction", "loss detection", and "security and content identification".

RTCP is designed to work in conjunction with RTP. In a RTP session, senders and receivers periodically send RTCP packets to provide flow and congestion control information. RTCP provides several packet types to carry control information including

sender and receiver reports which convey feedback of data delivery quality and information regarding membership. Therefore, conference participants may identify other members of conference through RTCP.

3.1.3 Ad-hoc Meeting

An ad-hoc meeting is a meeting set up on an ad-hoc network. Ad-hoc network is an extremely flexible and autonomous network dynamically and randomly deployed, has multi-hop topologies, and may be rapidly changing [29, 30]. Wireless nodes comprised in network architecture may become unpredictable at any time. Ad-hoc network is self-organizing to adapt demands of topologic changes. The network infrastructure is flexible and mobile and improvises itself as per node's instant activity. Communication between nodes is dynamically established because of network's adaptive control ability.

In an ad-hoc meeting mobile participants join the ad-hoc meeting in a random fashion. Ad-hoc meeting is not pre-arranged or scheduled. It is created dynamically. Participants may join or leave an ad-hoc meeting when they enter or leave the ad-hoc network. The state of the on-going meeting or conferencing may change rapidly due to unpredictable participant activities.

3.1.4 Key Elements in Ad-hoc Meeting Management

The dynamic nature of the ad-hoc network determines several issues involved in ad-hoc meeting management including routing optimization, security, and conferencing state maintenance. Since the environment of the SIP-based ad-hoc conferencing system under discussion is physically within a conferencing room, the issues regarding security and conferencing state maintenance are given importance in the thesis.

Ad-hoc meeting is different from traditional meeting. Any traditional meeting or conferencing is pre-scheduled and participants do not change (enter or leave) during conferencing. However, the dynamic nature of ad-hoc networks determines the on-going ad-hoc meetings' changes depending on mobility in users. Ad-hoc conference states are created dynamically according to certain standards and memberships are changed randomly. Users join or leave the conference when they join or leave the ad-hoc network. When all users leave the ad-hoc network the on-going conferencing comes to the end. The above discussion raises questions in ad-hoc meeting management like,

- How can users join a conference?
- How to authenticate conference participants?
- How to update the conference state?
- How to manage conference sessions?

3.2 SIP Applications and Related Works

SIP's usage in conferencing management can be better understood from the following introduction of SIP applications and SIP-based conferencing models. The following sections overview SIP applications and discusses several conferencing models where SIP is used for conferencing control. The proposed SIP-based ad-hoc conferencing system model stemming from these conferencing models will be discussed in section 3.3.

3.2.1 SIP Applications

Research works involving SIP's usage in different applications and environments are actively taking place. Noteworthy, "Supporting Mobility for Multimedia with SIP" introduces how SIP is used in mobility management [31] and "A Protocol for Wide-Area Secure Networked Appliance Communication" discusses SIP's usage in home networked appliance control [32]. Columbia University developed a multimedia conferencing server namely *sipconf* for deploying SIP-based Internet multimedia conferencing [33, 34, 35].

SIP in mobility management

The explosive growth of the Internet and "increasing demand for ubiquitous mobile wireless services" encourages design activities in all IP wireless networks that provide integrated data, voice, and multimedia services for roaming users [30]. Several forums, 3GPP, 3GPP2, and MWIF are active working groups on this field. All of these forums have selected Session Initiation Protocol (SIP) for session and mobility management.

This is mostly based on strengths of SIP such as simplicity, scalability, extensibility, and modularity. SIP is especially used for mobility support because it possesses the following advantages [31, 36].

- SIP allows user roaming depending on appliances rather than networks.
- By means of SIP address binding and registration, SIP provides route optimization and improves real-time services.
- SIP supports mobility at a level above IP terminal.

SIP supports terminal, service, and personal mobility on mobile Internet environments. Since SIP is an application layer protocol, it has the capability to provide an approach to support mobility on the application layer and is completely independent of lower layers. In SIP, a user needs to register to a SIP Registrar whenever his/her address changes. If a user registers several addresses, the SIP proxy forks the SIP INVITE to these addresses sequentially or in parallel until the call has been set up. The mechanism of binding users with their address is the foundation of SIP-mobility-support. For example, in the figure 3.3 shown below, user 'Bob' migrates to a foreign network. He needs to register his new address to his home proxy. When 'Alice' calls 'Bob', Bob's home proxy will forward Alice's request to his current address at the foreign network. In this way, Alice's call can always reach mobile user 'Bob'.

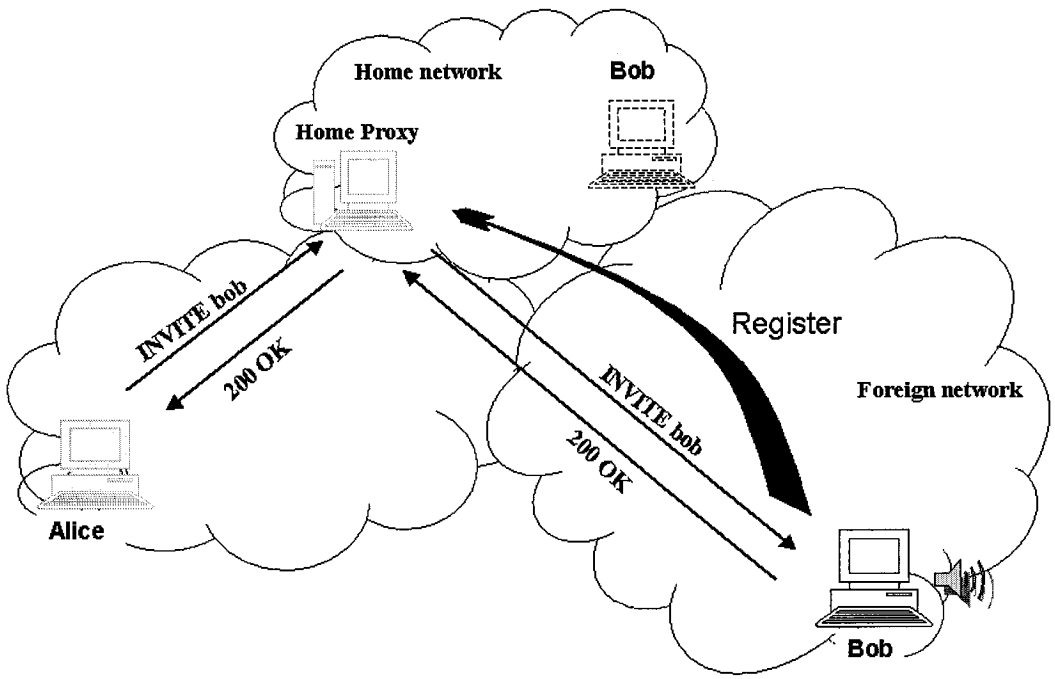


Figure 3.3 SIP in mobility management

SIP in home network appliance control

Using SIP in *home network appliance control* is another vital research topic. Since SIP supports session establishment across wide area networks, it provides an approach for users controlling their home appliances from the outside world [32].

In the example shown below in figure 3.4, all appliances have SIP UA's working on behalf of them. SIP UA's control these appliances through appliance control interfaces. The home owner sends commands to appliances via SIP requests from the outside world. These requests first go to the room proxy and the proxy forwards them to specific appliances and requested operations are executed 'successfully'. A firewall

coexists with the SIP proxy to prevent malicious calls getting through and makes sure the home network under consideration is secure.

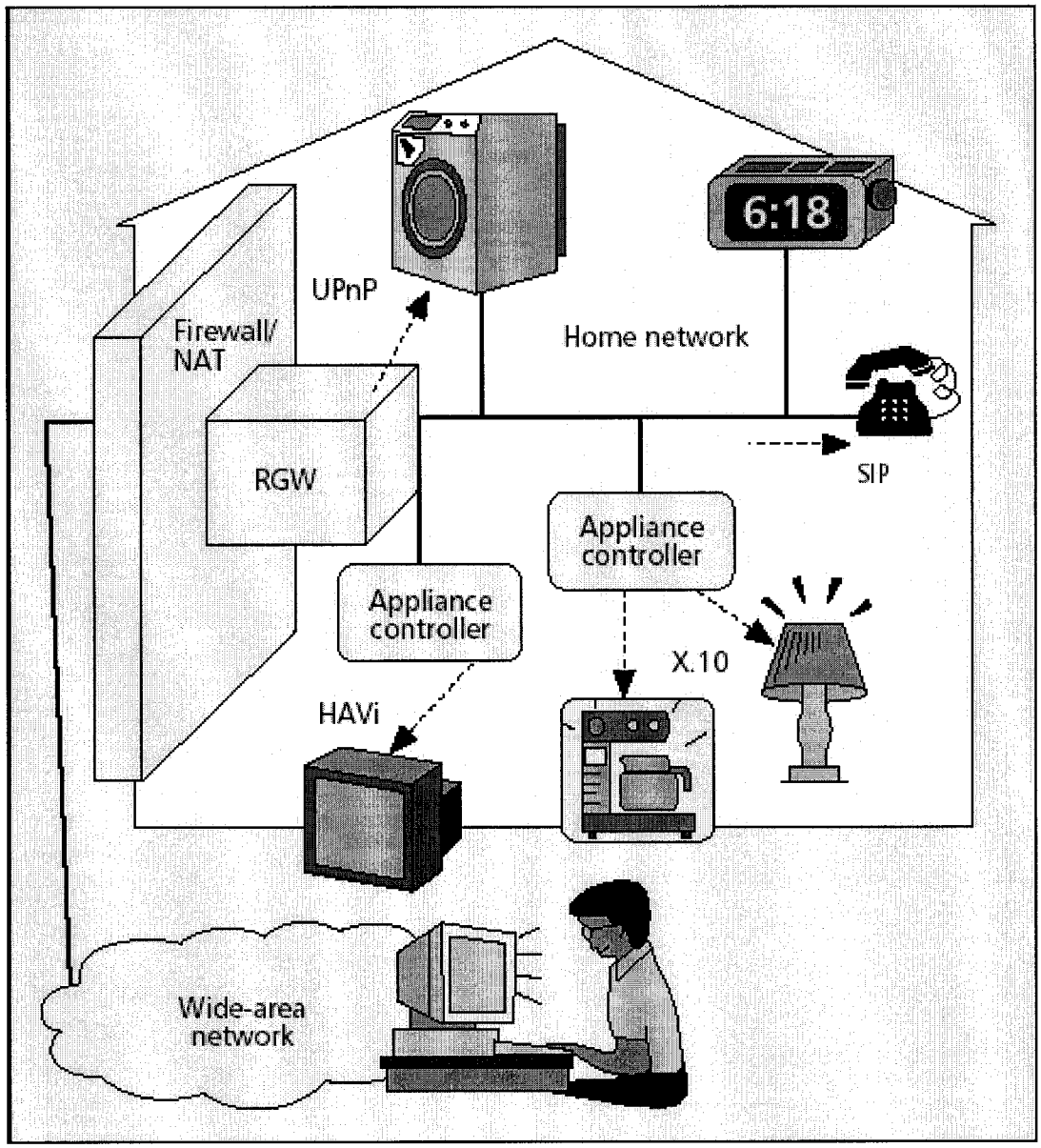


Figure 3.4 SIP in home network appliances control [32]

Internet conferencing control - sipconf

The multimedia conference server *sipconf* developed by Columbia University provides a centralized audio/video conferencing service by using SIP [33, 34, 35]. Sipconf provides a conferencing service system over the Internet. Users can communicate in live multimedia sessions. SIP is mainly used to manage and control multimedia sessions between users and the conference server. The architecture is designed using the client/server approach as shown below in figure 3.5

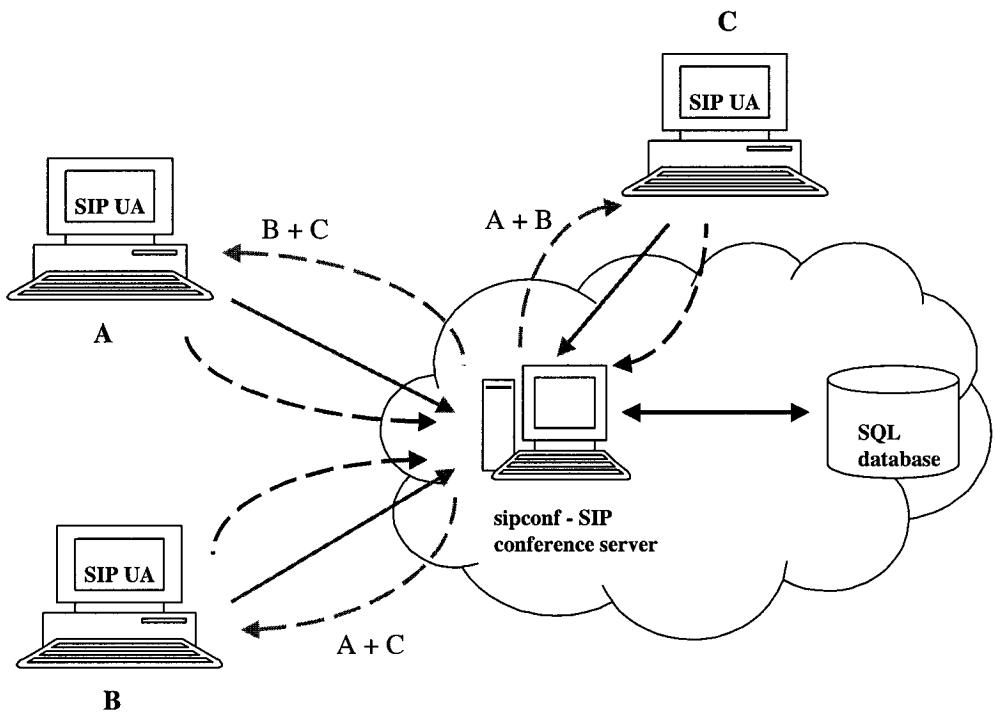


Figure 3.5 The *sipconf* application

In this dial-in centralized conferencing system, a conference server processes both signaling and media mixing. For the signaling part, a user sends SIP requests to the

conference server to join or leave a multimedia conference through the Internet. Every conference has a unique identifier to distinguish itself from other conferences. As long as a user knows the SIP URL of a conference, he/she can join the conference through a SIP request. Users join a specific conference by indicating the corresponding conference ID in SIP requests sent to the conference server. The conference server is also responsible for "mixing and redistribution of media streams" [34]. The conference server receives RTP media streams from all participants and sends mixed streams back to participants. The conference data, such as conference schedule, identifier and authentication mechanism for participants, is stored in a SQL database and can be updated from a web page.

In all of the above applications, SIP is used mainly for locating destinations and establishing corresponding sessions between callers and destinations. However, in a conferencing room under consideration, users know each other without involving SIP. Therefore, instead of locating the destination, the utility of SIP in this thesis work should focus on managing and controlling communications between conference participants as well as participants and resources in the room.

The conferencing service provided by *sipconf* application is for Internet users and it is completely different with the context under consideration in this thesis. The context considered in the thesis is a conferencing room in which more resources are provided to users rather than a conferencing service only. Therefore, the thesis should give

importance to control and manage resource accessibilities in the room as well as the conferencing service. An autonomous system should represent the room entity to manage all activities due to context changes in the room and the management needs to be deployed according to room policies. The agent approach effectively fulfills the requirements for such a complex situation when compared to the client/server approach. Agents help to form an autonomous and intelligent system by performing tasks on behalf of resources or users. Each agent performs one specific task and cooperates with other agents to manage conferencing activities autonomously. Also the concept of agent mobility can be used to improve system performance. Mobile agents could move from one location to another location and execute tasks in the new foreign locations. These mobile agents enrich system capability without increasing the workload of the system. For example, when a user joins a conference, a new service agent can be dispatched to the user by the system so that the new service can be provided to the conference participant.

3.2.2 Models of SIP-based Multi Party Conferences

A suitable conferencing model should be selected in order to design a conferencing system with SIP. SIP is originally designed as a large-scale multiparty conferencing protocol and it is the most important SIP application. "Models for Multi Party Conferencing" and "Signaling for Internet Telephony" state the usages of SIP in multi party conferencing, which can be generalized in six different models [37, 38]. By examining the existing models, we proposed the SIP-based ad hoc conferencing model.

3.2.2.1 End-Mixing System

An end-mixing system is fundamentally based on the 3-party conferencing system. A user works as a mixer mixing media streams from two users and sending back the mixed stream to these two users. In this way all three parties exchange information.

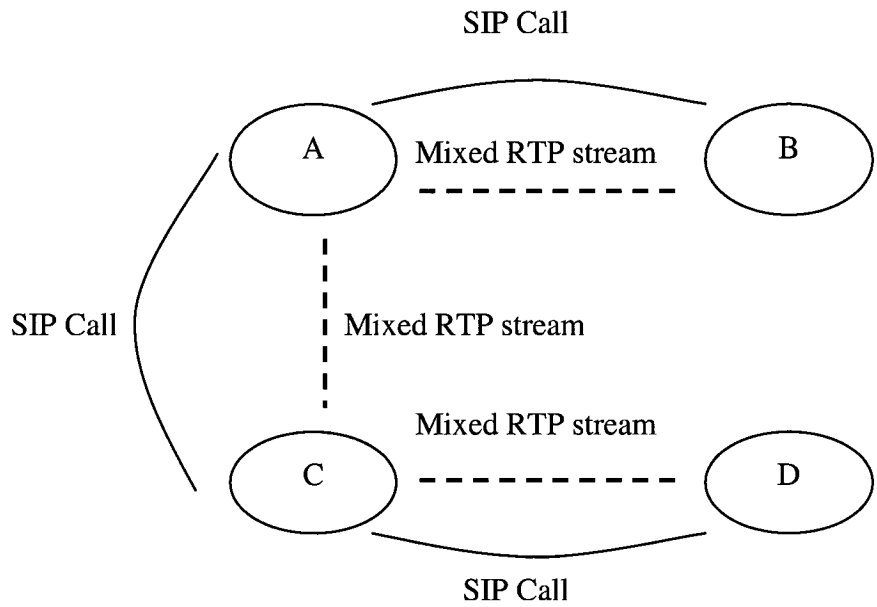


Figure 3.6 End System Mixing model

As shown above in figure 3.6, four SIP UA's are running for four users 'A', 'B', 'C', and 'D'. First, UAs established a session between user 'A' and 'B' through standard SIP INVITE. Then user 'A' creates another session between 'A' and 'C'. The stream 'A' sent to 'B' contains 'C's stream and the stream 'A' sent to 'C' contains 'B's stream. 'C' may invite 'D' to join the conference by sending a SIP INVITE. The stream 'C' sent to 'D' contains both 'A's and 'B's stream and 'B' can get both 'C's and 'D's stream from 'A'. Any conference participant can invite other users to join the conference as long as

they have the capability to mix media streams. Conference participants know others' identities through RTP.

This model has several drawbacks. Firstly, the conference completely depends on the mixer. When the mixing SIP UA leaves the conference, the conferencing would come to a stop. Another issue is due to the increase in number of conference participants, which results in bandwidth and loop problems. This model is not suitable for large-scale conference.

3.2.2.2 Large-Scale Multicast Conference

A large-scale multicast conference usually is a pre-arranged conference with allocated multicast address. Users join a conference by joining one multicast group and send media to this group for information exchange. All conference participants use the same conference multicast address and port number for communication.

SIP is used to inform conference participants' conference multicast addresses. In the figure 3.7 shown below, when user 'A' sends a SIP INVITE to user 'B', user 'A' also gives complete information about the on-going conferencing in the SDP of a SIP INVITE. From SIP INVITE's SDP, user 'B' gets all required information to joining the particular conference. Conferencing participant may invite any user to join the conference by sending full conference description in the SDP. In the conference, participants learn other members' identity through RTCP.

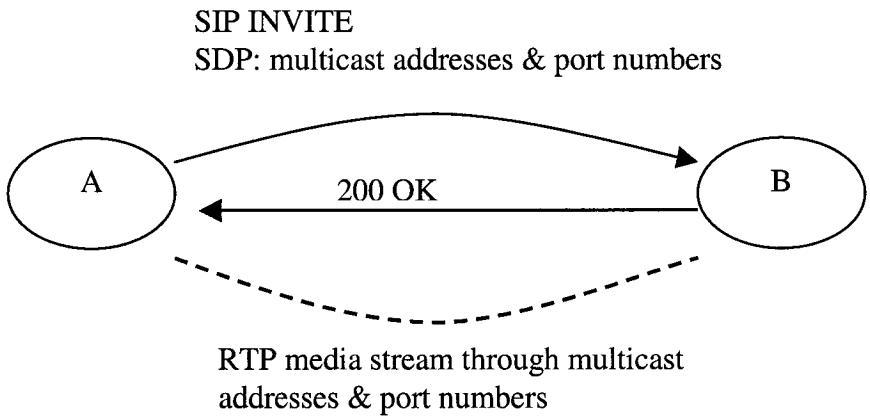


Figure 3.7 Large-scale Multicast Conference model

3.2.2.3 Dial-In Conference

The dial-in conference model is different from above discussed models since it has a conference server responsible for all conference service logics. This conference server has point-to-point SIP and RTP relationship with every conferencing user. There may be more than one on-going conference at a certain time. Conference ID or SIP URL identifies a particular conference. The server works atop of a normal SIP UA. Users call the server using SIP INVITE and the conference server allows users to join the conference by sending back "200 OK". The server receives media streams from conference participants, mixes them, and separately sends out mixed stream to all individuals. Since the service logic is present in the server, it has overall control over the conference. The server can enforce conference policies such as membership authentication and conference number limitation.

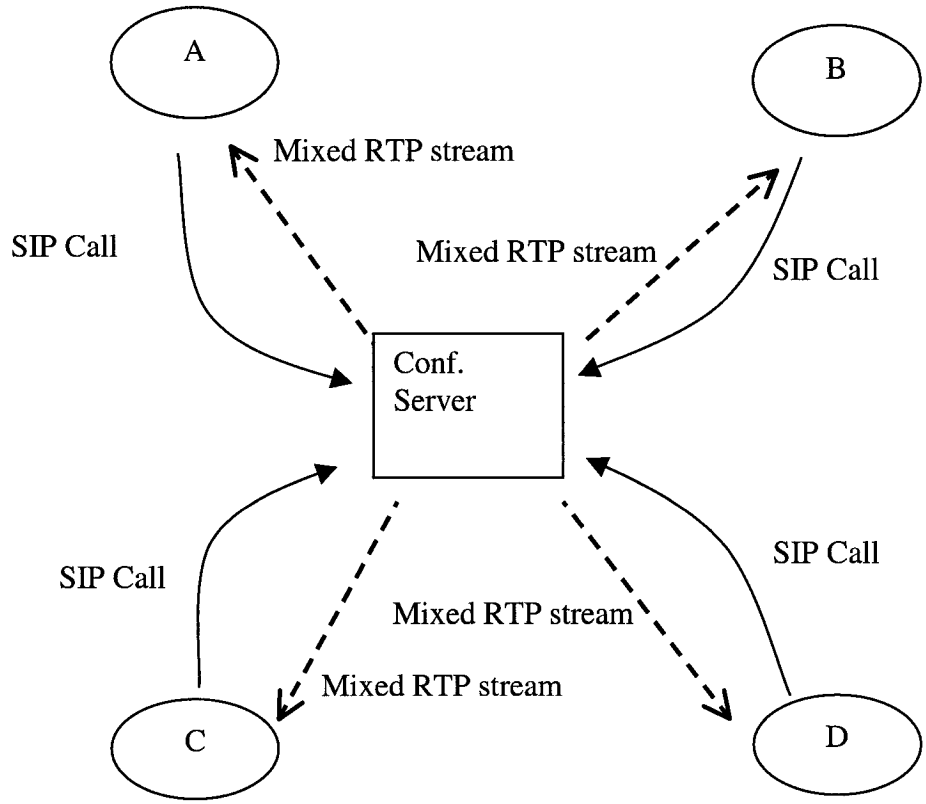


Figure 3.8 Dial-In Conference model

Conference participants can invite other users to join the conference by sending a SIP REFER [appendix A.1] request. In REFER request, the 'inviting' party informs 'invited' party about conference server address and conference ID. This REFER request causes invited user sending a SIP INVITE with indicated ID to conference server in order to join the conference. Conference participants learn about other users through RTP.

The conference scalability of this model depends on bandwidth and processing capability of conference server. The performance may decrease because of bandwidth bottleneck or servers' low processing power.

3.2.2.4 Ad-hoc Centralized conference

In an ad-hoc conference, two users for example 'A' and B could start a call with SIP. After the session is established, one user (say 'A') takes the responsibility to transmit the session to a conference server. To do so, 'A' tries to discover a conference server supporting ad-hoc conferencing and chooses a conference ID or SIP URL for this particular conference. User 'A' sends a SIP INVITE to the conference URL and this causes the conference state to be created in the conference server. Then, user 'A' sends REFER to user 'B' to join the conference. The remaining operations of the conference server are identical to dial-in conference model operations.

Inviting other users to join the conference is similar with user 'A' inviting user 'B' to join the conference. A caller send a REFER to the invited user and this causes the invited user to send a SIP INVITE to the conferencing server. Conference participants learn about other users' identities through RTCP. Similar to a dial-in conference, the performance of ad-hoc conference depends on bandwidth and server processing power.

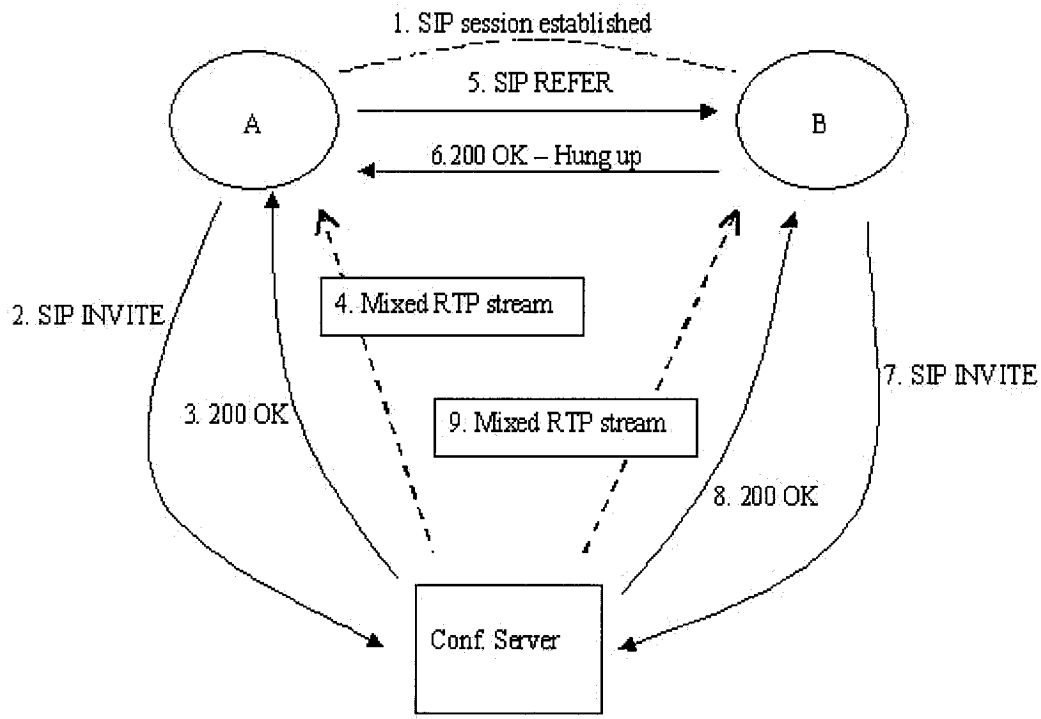


Figure 3.9 Ad-hoc Conference model

3.2.2.5 Dial-Out Conference

The dial-out conference is very similar to the dial-in conference. The only difference is that the conference server makes decisions as of which users can join the conference rather than users themselves. The conference server chooses conference members and sends them SIP INVITE to establish sessions. Once users accept invitations, the conference sessions establish.

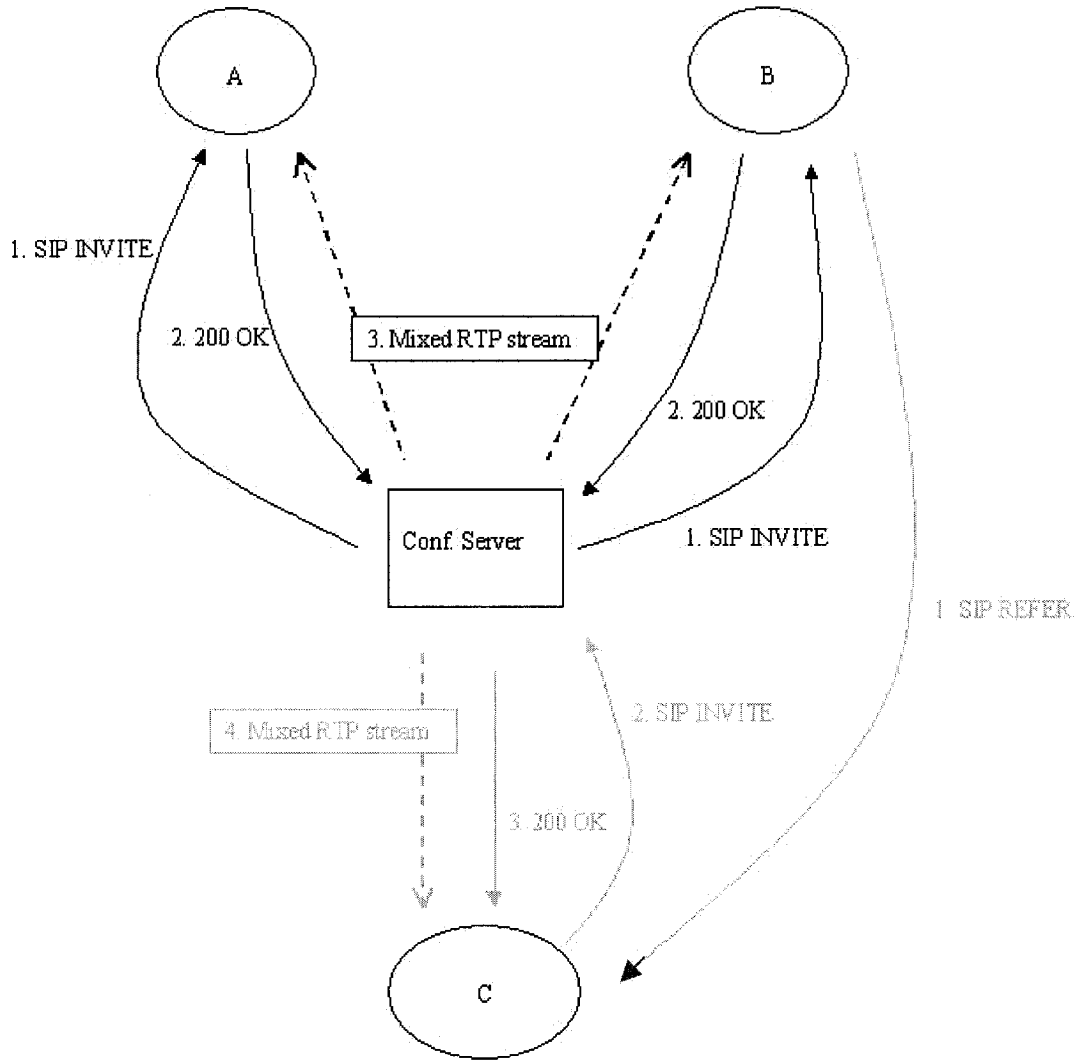


Figure 3.10 Dial-Out Conference model

If the conference server supports incoming SIP INVITE, other users may join a dial-out conference as they do in a dial-in conference. As discussed before, SIP INVITE indicates caller's address through message header FROM. As seen above in figure 3.10, when user 'B' receives an INVITE from the dial-out conference server, user 'B' knows the conference URL from FROM. If user 'B' wants to invite user 'C' to join the conference, he/she can just sends a REFER to user 'C' which indicates conference URL.

This enables user 'C' to send a SIP INVITE to specific dial-out conferencing server to join the conference. A dial-in-conference supporting server accepts calls and thus approves 'join' requests. In a dial-out conference, users are aware of other participants through RTCP.

3.2.2.6 De-central Conference

The de-central conference is named so because of centralized signaling and distributed media. Unlike dial-in and dial-out conferences, the conference server in de-central conference only handles signaling but not media. Media is sent directly between conference participants through multicast or "multi-unicast" (*a user sends multiple packets addressed to each recipient*) [37].

In De-central model, the job of conference server is based on 3PCC (third party call control) [Appendix A.2] of SIP. While working as a 3PCC controller, conference server negotiates with all conference participants to allocate a new media line for their communication with new users. This will cause each participant to assign a new IP address and port number for the communication. Negotiation is done through SIP. After this, user 'A' joins the conference and can send/receive media streams to/from each participant. Participants discover other users through RTCP.

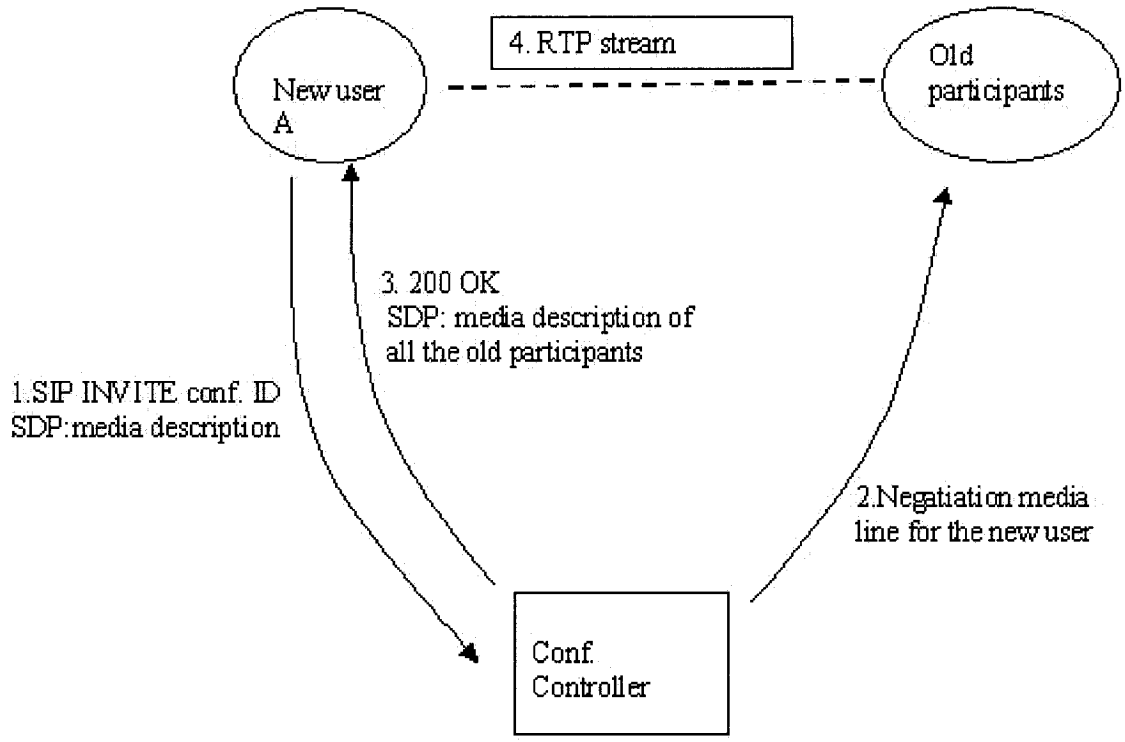


Figure 3.11 De-central Conference model

In the de-central conference model a point-to-point SIP relationship exists between all participants. Because of the fact that this models' work is based on SIP 3PCC in which a lot of SIP INVITE transactions are involved, it can be seen that this model is suitable for situations where bandwidth is greater and PC's are more powerful.

3.3 SIP-based Ad-hoc Meeting Management Model

3.3.1 Why SIP

When designing the conferencing system architecture, considerations need to be made regarding flexibility, extensibility and scalability [39], for example, the extent to which

the conferencing model may be applied and the flexibility level of functionalities should be known. To realize this functionality, what kinds of policies should be provided. Extensibility, namely, how far the architecture can be extended to integrate new functionality is an important issue.

As discussed before, there are several solutions for managing Internet multi-party conferences. H.32x-based solution and SIP-based solution are note-worthy. H.32x [27, 40] is an International Telecommunication Union (ITU) standard widely used for multimedia communication. H.32x is more concerned about interoperability with PSTN (Public Switched Telephone Network) than with the Internet. And this makes H.32x less adaptive in the Internet communication. When used in the Internet multimedia conferencing, H.32x conferencing model usually has tight centralized control structure. This kind of structure is hard to be extended and poorly supports changes in a running conference and reconfiguration of the conference.

SIP-based solution is chosen for the conferencing management model under discussion, because SIP is more adaptive towards the Internet communication. There are several conferencing models already described based on SIP. Either one of these conferencing models or an integrated model can be adopted to meet specific environment requirements in the thesis. Also, SIP doesn't have any enforcement towards conference members and hence participants can negotiate media types during session establishment. Even after session establishment, users can still negotiate to change their media type.

These properties makes the discussed system architecture flexible and expandable, so as to integrate other media types for conference communication.

SIP extension also makes the discussed conference architecture an open system. SIP extension can add new functionalities to the system. Since SIP is a signaling protocol proposed for the Internet communication, the conferencing model can easily be adapted from single conferencing sites to multi-conference sites over the Internet.

3.3.2 System Description

This SIP-based ad-hoc conferencing system is designed in a way so as to be capable of managing ad-hoc conferencing for mobile users in an ad-hoc environment. The system aims to provide an adaptable conferencing application. There are several distributed conference sites over the Internet. Users can enter into any conference site and could automatically join the ad-hoc conferencing. The plan to design application architecture for multi-site conferencing system is to first start from single-site environment, i.e., one conferencing room. After designing architecture and scenarios for one conference site, the system can easily be expanded to support larger and more complex environments, namely, multi-site conferencing over Internet. The goal is to minimize the complexity of the appearance of the conferencing system and to make it autonomous and prompt to any change in conferencing circumstances.

The strategy followed in the thesis is based on combined use of SIP and agent technology. The former is used to initiate and manage ad-hoc conferencing and the latter is used to minimize user manipulation. Agents work on behalf of the conferencing room entity to control the overall system and services, especially, the *conference agent* manage and control all conferencing activities in the room. Using a precisely defined procedure, the conferencing system mechanism is designed and the approach is simulated in prototype implementation. Results are very encouraging as the approach clearly helps to obtain an integrated autonomous conferencing system with agents collaborating and fulfilling their own goals in a systematic way.

The SIP-based ad-hoc meeting model encompasses characteristics of several conference models mentioned before. Firstly, the SIP-based ad-hoc conferencing model is a dial-out conference type. As in a dial-out Conference, there is a *conference agent* acting as centralized controller in our model. However, the *conference agent* in this SIP-based ad-hoc conferencing system isn't responsible for mixing media streams. Instead, the *conference agent* decides the multicast address for the conferencing and notifies conference information to users through SDP of SIP INVITE. The communication of participants is similar to large-scale multicast conference model wherein all parties use same multicast address and port number for information exchange. Moreover, the model presented in the thesis has an ad-hoc aspect due to the fact that the conferencing state is dynamic as the user enters or leaves the conference room, although its operation mode is completely different with the existing ad-hoc conferencing model introduced in section 3. Moreover, though the *conference agent* doesn't work based on 3PCC as in the De-central

conference model, the SIP-based ad-hoc conferencing model does show characteristics of centralized signaling and distributed media. The *conference agent* centrally handles SIP signaling issues, but the media is sent directly between conference participants through multicast.

SIP can notify users about conference addresses and can provide assistance for negotiating policy-matching media types such as, video or audio. SIP can also assist in initiating conference and managing conference sessions. Agents facilitate communications between conference participants and manage resources in the conferencing room. The room entity consisting of a group of agents dynamically maintains the conference state and controls 'inviting' users. Efforts are made to,

- Define architecture to fully embody the ad-hoc conferencing proposal.
- Give generic scenarios for conferencing management and facilitate communications between conference participants.
- Manage static and dynamic resources of the room and control access rights of resources.

The overall procedure followed is elaborated as follows. When users enter a conferencing room, the room entity will check their identifications. If a user is authenticated, he/she automatically joins the conference and could simultaneous access services and devices in the room (depend on user profile and maintained policies). In addition to local resources, users grant access to resources and services available on their own platforms ('brought in' services). Such access is performed in accordance with

users' policies as well as room enterprise policies. A sub-group of participants could initiate a sidebar session for a private meeting while still participating in the main conference. Users can define policies for sidebars initiated by them. The system also provides service for users identifying other participants through SIP presentation service rather than RTP or RTCP.

3.4 Summary

This chapter discussed related techniques in detail. The multimedia conference, media transportation, and ad-hoc conference were introduced briefly. This was followed by SIP and its application descriptions, especially various SIP-based conferencing models were discussed in detail. Finally, reasons to choose SIP for the conferencing management were stressed.

Next chapter describes the SIP-based ad-hoc conferencing model in detail. Architecture and scenarios are fully addressed. Features of the conferencing system and employed mechanisms are discussed in design description.

Chapter 4 Architecture and Features

This chapter first introduces the proposed architecture of the SIP-based ad-hoc meeting management system and then elaborates system design. As discussed before the goal of this system is to design a conferencing management system that could be applied in multi-conferencing sites over the Internet. To narrow down the problem domain, the current focus is on solving problems in a single Internet site.

Approaches to provide various services to conference participants in one conferencing room are discussed in detail. The system presented in the thesis provides an approach that combines agent technology and SIP to manage an ad-hoc conference. By using agent technology and SIP, the room entity could dynamically create and manage an ad-hoc conferencing system. The proposed architecture applies to multimedia conferencing also, including audio and video. The features and services of SIP-based ad-hoc meeting management system discussed in the thesis are adapted to one conferencing room. However, based on these scenarios, the system can be extended to support multiple sites and could adapt to its 'current' Internet environment.

4.1 Introduction

The physical context of the SIP-based ad-hoc meeting management system is a conferencing room. When users enter a conferencing room to take part in an on-going conference, they will be automatically invited by the room entity to join the on-going conference via SIP. The conferencing room under consideration has devices too, which could provide various services to conference participants. However, there are certain enterprise constraints to authorize users to join the on-going conferencing and to utilize available services. The conferencing will come to an end when all users leave the conferencing room.

Two types of conferences could take place in the conferencing room. The first type is the main ad-hoc conference. The main ad-hoc conference should include all authenticated users in the room and exists as long as conference users stay in the conferencing room. The other type of conference is sidebar conference, wherein a subsets of participants involve in a private conference. Both main ad-hoc conference and sidebar conference have unique SIP addresses to identify them. Different conferences have different conference IDs. The conference ID may be addressed using an URI (Uniform Resource Identification) in the form of <conference ID>.adhoc@room.com [41].

The 'intended' conference participants can also bring their devices or services along with them while entering the conferencing room. Users offering services can state preferences for their services so as to minimize or maximize their service accessibility. Because sidebars are private sessions initiated by conference users, these sessions can be

controlled by users themselves who initiate it. For example, users could define policies for their sidebars.

4.2 Entities in the System

Various agents collaborate with each other in the conferencing system and work on behalf of the room entity. Collaboration between agents is essential to make the whole system run properly. Though the thesis only focuses on the conferencing part, a brief overview of all agents involved in the overall system is given below for better reader understanding.

Context Agent (CTA): The Context Agent is responsible for detecting conferencing room activities. The Context Agent captures information such as, different type of devices that are in existence in the room, users entering and leaving the room, etc. These captured information are provided to higher application levels. The Context Agent also works as the Presence Agent, i.e., it is responsible for processing incoming SUBSCRIBE requests and gives back corresponding notifications. For example, when a user enters or leaves the room, CTA captures this change, creates a NOTIFY, and finally passes the notification to other subscribers. Here the subscriber is RM of the room (introduced below).

Policy Agent (PA): The Policy Agent maintains systems' policies. Enterprise constraints are in existence for all conference participants, and user-defined policies for private devices and services. While enterprise constraints represent requirements of the

enterprise, user-defined policies are indicated directly by users. Example policies include conference scheduling, limited conference participants, specific conference participant media type, etc. In regard to different requirements of enterprises and users, policies can easily be modified. Thus, the behavior of the system can be modified without influencing the system architecture.

Room Manager (RM): The Room Manager is in charge of the room at the highest level and is aware of everything happening in the conferencing room. For example, RM is aware of users entering and leaving the room. RM is also responsible for creating and terminating conference states in a conferencing room. It decides when to create a conference state and when to terminate it. RM also does SIP registration to SIP servers on behalf of all users and devices in the room in order to intercept SIP requests. Hence, all SIP requests are forced to pass through RM. In this way, RM takes overall control of conferencing room communications.

RM works on top of SIP B2BUA, i.e., it acts as the User Agent Server (UAS) when it receives a SIP request. Then, it processes the received request according to internal functions. Depending on results obtained, RM either refuses the request or accepts the request. Later it acts as the User Agent Client (UAC) and generates a new request to the 'intended' destination.

User Personal Agent (UPA): The User Personal Agent (UPA) is a service agent which runs on users' local platform and processes users' interaction with other agents in

an 'on-going' conference. It handles issues such as registering users' personal policies for their devices, initiating service-accessing requests for users, etc.

Other service agents: There are various service agents 'alive' in the conferencing room under consideration and each one works on behalf of one special service, such as sidebar agent, printer agent, etc. These agents are responsible for controlling and maintaining services and processing requests received by 'active' services in the room.

4.3 The Main Conference

The main conference is an ad-hoc conference dynamically created and managed by RM. Users register with the association based on entering the room. When RM receives information regarding users' appearance in a conferencing room from lower context awareness level, it automatically starts a new conference.

4.3.1 Setting Up and Joining the Main Conference

RM subscribes to the Context Agent regarding users' presence. When users either enter or leave the room, the Context Agent will notify these activities to RM. The notifications would also include users' profiles. When the first user enters the room, RM automatically creates a conference state and invites the user to join the conference.

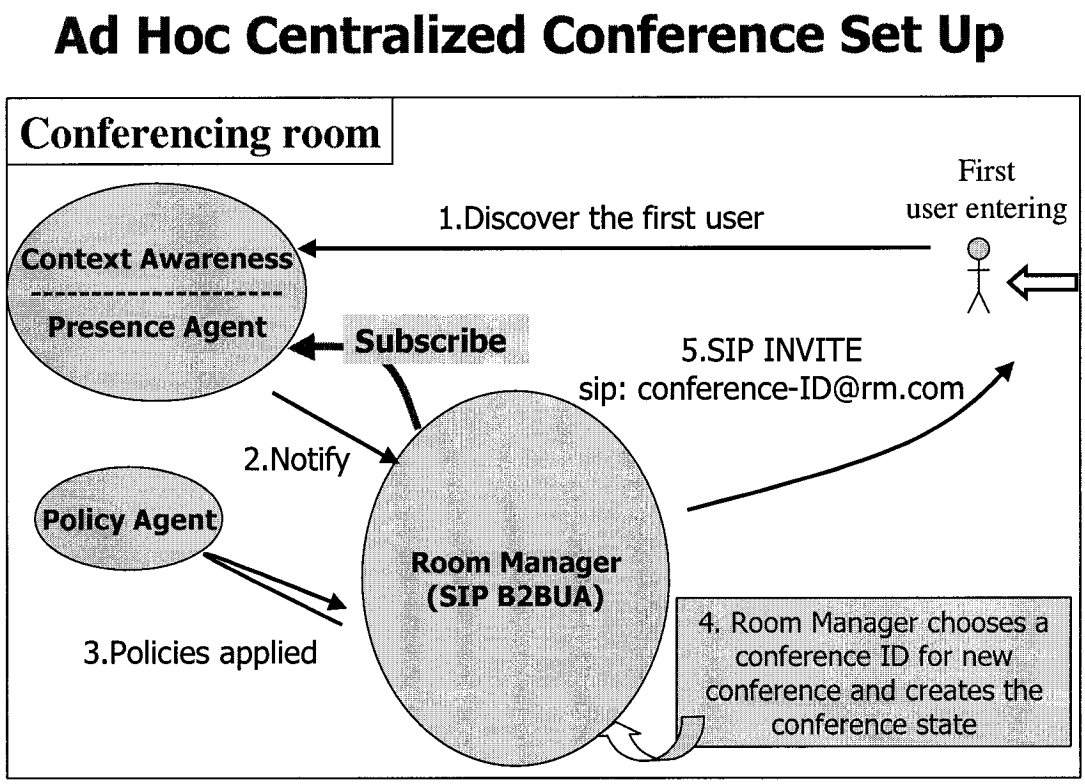


Figure 4.1 Start up the ad hoc conference

As seen from figure 4.1 shown above, whenever RM receives a notification stating new user arrival, it authorizes the particular user to use 'active' services matching his/her profile. RM first communicates with the Policy Agent to obtain the user's priorities and 'rights'. After the user has been successfully authenticated, RM will choose a conference ID for this main conference and create the ad-hoc conference state. At the same time, RM makes sure that the enterprise or obligatory policies are also applied. For instance, an enterprise policy may include *particular communication media types, number of conference participants allowed, etc.* After this, RM sends a SIP INVITE to the particular user under consideration to join the main conference. The SIP invitation

would include conference session descriptions, such as policy-specified media types for communication. The user may then join the conference if he/she agrees on the media description and accepts the received invitation.

Similar procedure is followed to invite all other users too, except that the conference state would have already been created. Also it is worth to note that policy plays an important role to determine the behavior of the system. As discussed earlier, policies determine whether or not a user can join a conference (both new and on-going). For example, when a user enters the room, RM first checks whether maximum number of conference participants is already reached before sending an INVITE. If the maximum limit has already been reached, the user will still be rejected even though he/she has proper authorization. These rejected users can't access any services in the room, even though they are 'actively' present.

4.3.2 Leaving the Main Conference

Users automatically quit conferencing when they leave the conferencing room. When a user leaves the room, the Context Agent gives a user-leaving notification to RM. After RM receives the notification, it automatically 'removes' the particular user from conference user list by sending a SIP BYE.

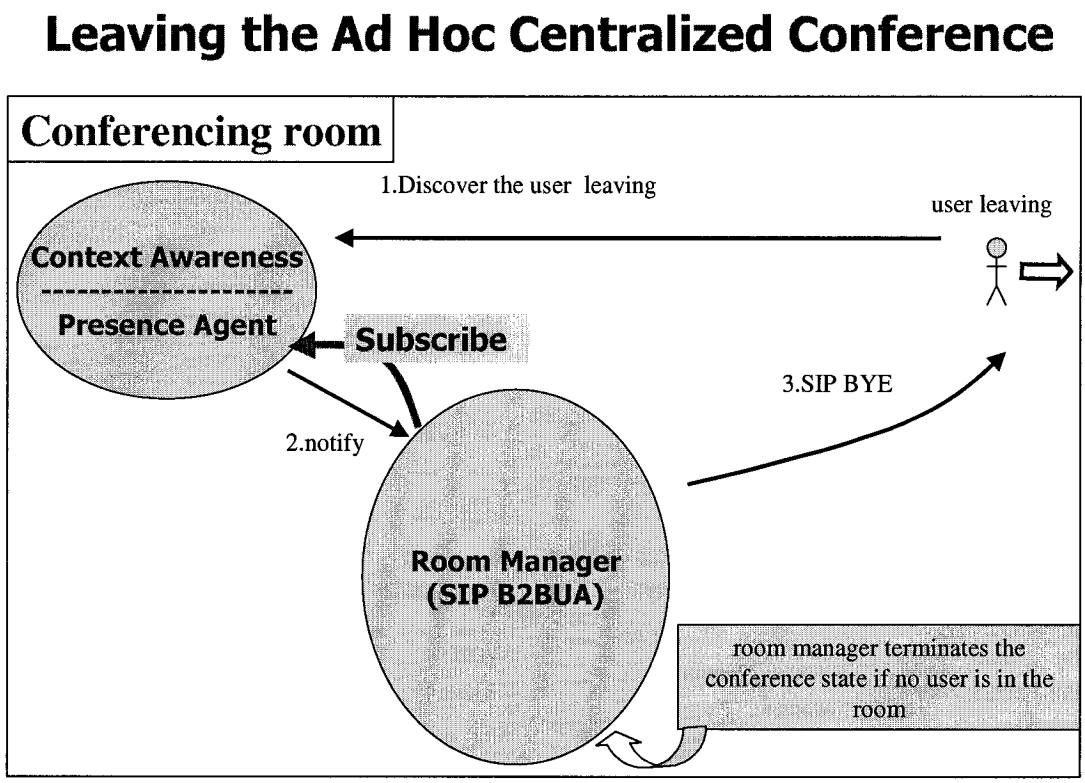


Figure 4.2 Leaving the ad hoc conference

If all users leave the room, RM determines that the conference is over and terminates the conference state consequently. Later, RM notifies all agents involved in the system to clear all data related to this conference. Figure 4.2 depicts how the system behaves during above discussed situation.

4.4 Source-policy

After users join the conference, they may utilize devices and services in conferencing room depending upon their profiles. Users are also allowed to bring their own devices or services into the conferencing room. For these private services, users may have their own

policies or preferences to control service accessibility. These preferences are called 'source-policy' and this section further discusses about these source-policies.

4.4.1 Defining the Source-policy

Users can define their private policies in the form of Call Processing Language (CPL) script [7, 8]. The Call Processing Language is an XML-based language that has been used to describe and control Internet telephony services. The intention of CPL is to give Internet telephony users a powerful language to describe their services and features. It is suitable to be used in situations where non-trusted users are allowed to develop services.

SIP requests contain message headers and optionally contain message bodies as discussed earlier. In the message bodies, either there are session descriptions or other service descriptions, such as a XML script. "Transporting user control information in SIP REGISTER payloads" suggests transporting user control information such as CPL scripts as payloads of a SIP REGISTER to the Internet telephony server and executing these scripts there to apply users' preference in the Internet telephony [42]. The thesis proposes using CPL for describing users' private policy in the ad-hoc conferencing management system. Conference participants define their source-policy in CPL scripts and upload these scripts to room entity through the SIP REGISTER. The Policy Agent extracts the policy scripts from REGISTERs and stores them in 'policy storage' for future look up. When needed, RM could retrieve information from PA and makes sure that policies are

enforced to control system behavior. As a result, user-defined policies and preferences are applied.

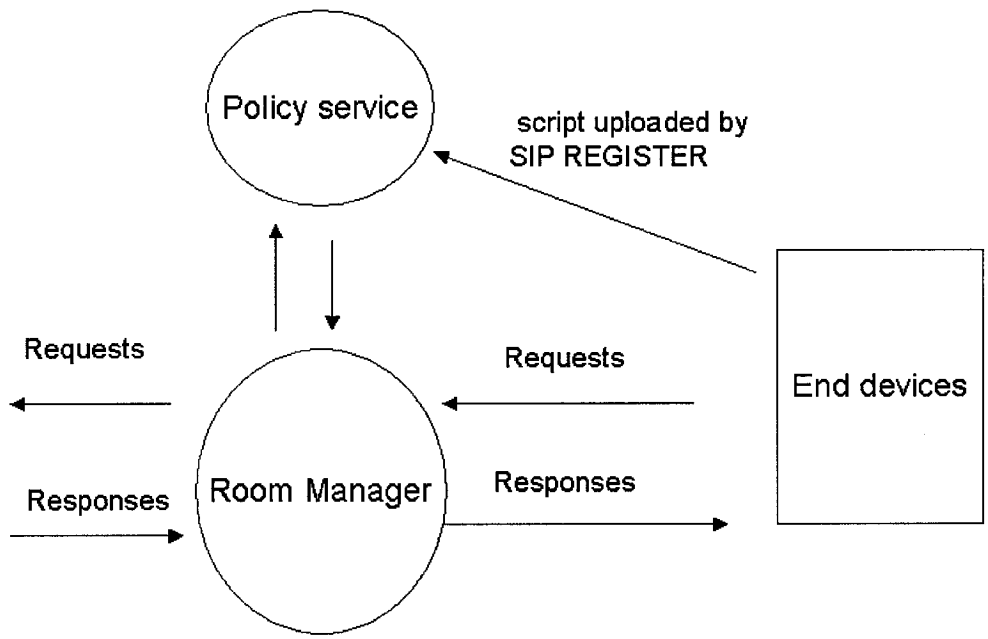


Figure 4.3 Source policy mechanism in CPL

4.4.2 Source-policy in Conferencing Room

Users who bring their own devices into the room have to register their private policies to the room entity in regard of 'brought in' devices. Understandably RM doesn't want to expose details of room service information to users while allowing users to fully describe their private policies.

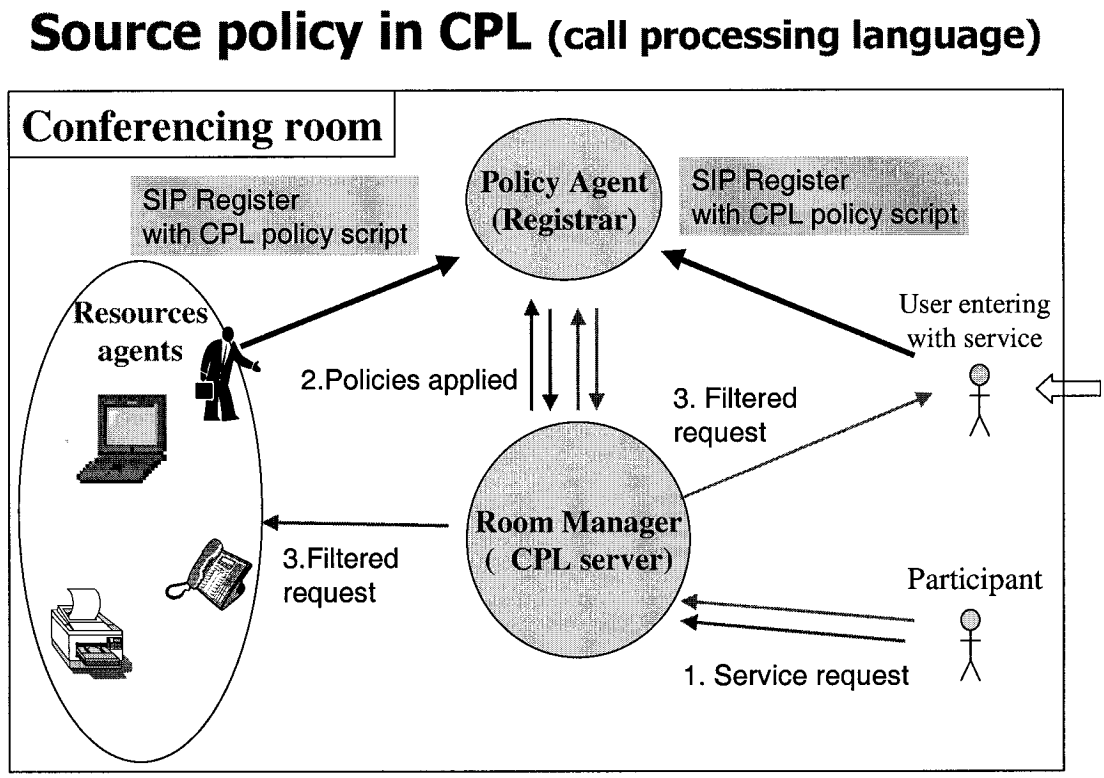


Figure 4.4 Defining the source policy in CPL

First, users have to register their private constraints as source policies as shown above in figure 4.4. After users define their policy scripts, these scripts have to be uploaded to the server so as to be 'active'. Users send policy scripts to the SIP Registrar in the body of a REGISTER request. The SIP Registrar processes the received registration request. From the Registrar, PA extracts the script, validates it, and updates the device's source policy.

Devices provided by conference room entity also have their own policies to control accessibility. Users with different privileges can access different types of services. RM distributes corresponding access rights to entering users according to

device's policies. There are various resource agents working on behalf of these devices in the room. If any 'device' policy changes, its service agent will register the new policy script to PA in the same way as users register their private source policies.

The user-defined source policy hibernates first and could be activated later. As discussed before, RM has the highest authority and is aware of all happenings in the room and all requests in the room pass through it. If a user wants to use any service available in the room, resources of the room or private devices, he/she sends a service request to the particular service or device. When the request passes through RM, it will check the policy agent for both enterprise obligatory policy and private source policy. If RM finds source policies, it will act as a CPL server to execute CPL policy scripts. If the request contradicts with the source policies, it will not be considered for further processing. Only policy-filtered requests are forwarded to destination services or devices as showed in figure 4.4. After the request is approved, the user can access the requested service.

The source policy is stored in PA as long as the ad-hoc main conferencing remains active. When all users leave the conferencing room, RM ends the on-going conferencing and notifies PA to clear all user-defined source policies. The call flow is showed below in figure 4.5.

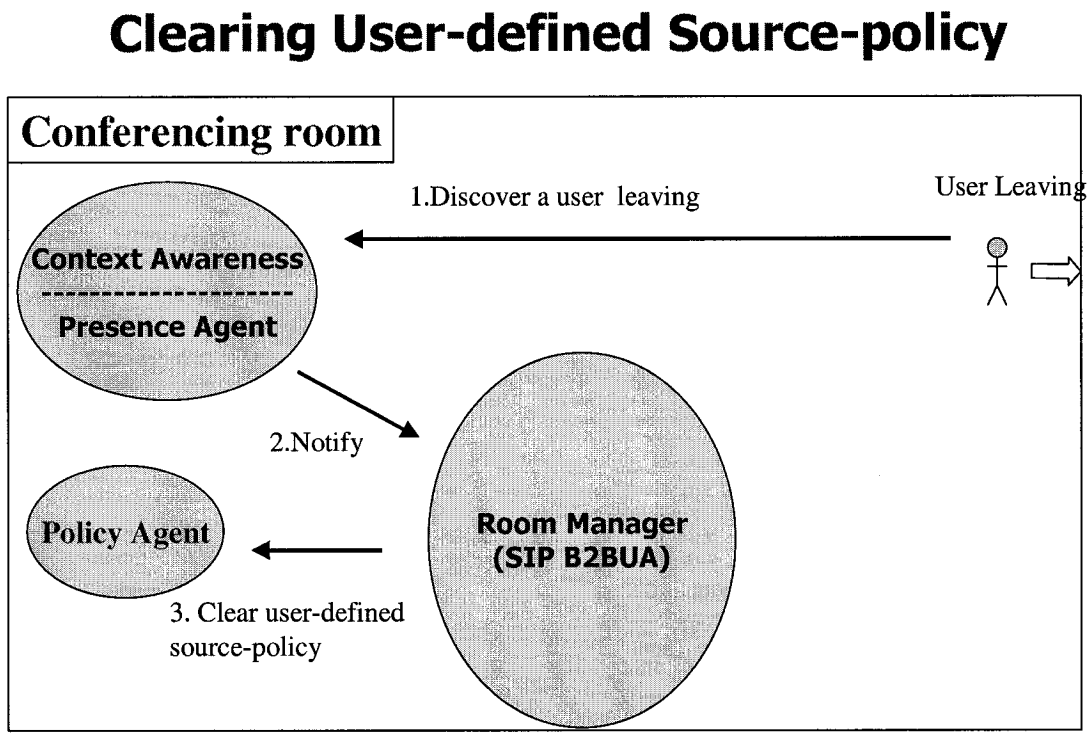


Figure 4.5 Clearing source-policy

4.5 Sidebar Association

A sidebar association is a private conference started by conference room participants for the purpose of sharing private opinions including topics related to an on-going main conference. A sidebar is a conference with the control on the user part. It is more like a scheduled conference. Users may define policies for sidebars, such as scheduled conference time, maximum number of participants allowed, sidebar media types, etc. Sidebar initiators can also decide who may join their sidebar and who may not. A sidebar

initiator chooses a new conference ID for a new sidebar and sends a SIP INVITE to the called party to initiate the sidebar session.

4.5.1 Sidebar Set Up

Before setting up a sidebar, sidebar initiators need to register their sidebar policies so that sidebar requests can be filtered by these sidebar policies. Similar to defining source policies, users specify their sidebar policies with CPL language and upload them to the PA using SIP REGISTER as shown below in figure 4.6. RM will work as a CPL server to execute sidebar policies to process future sidebar requests.

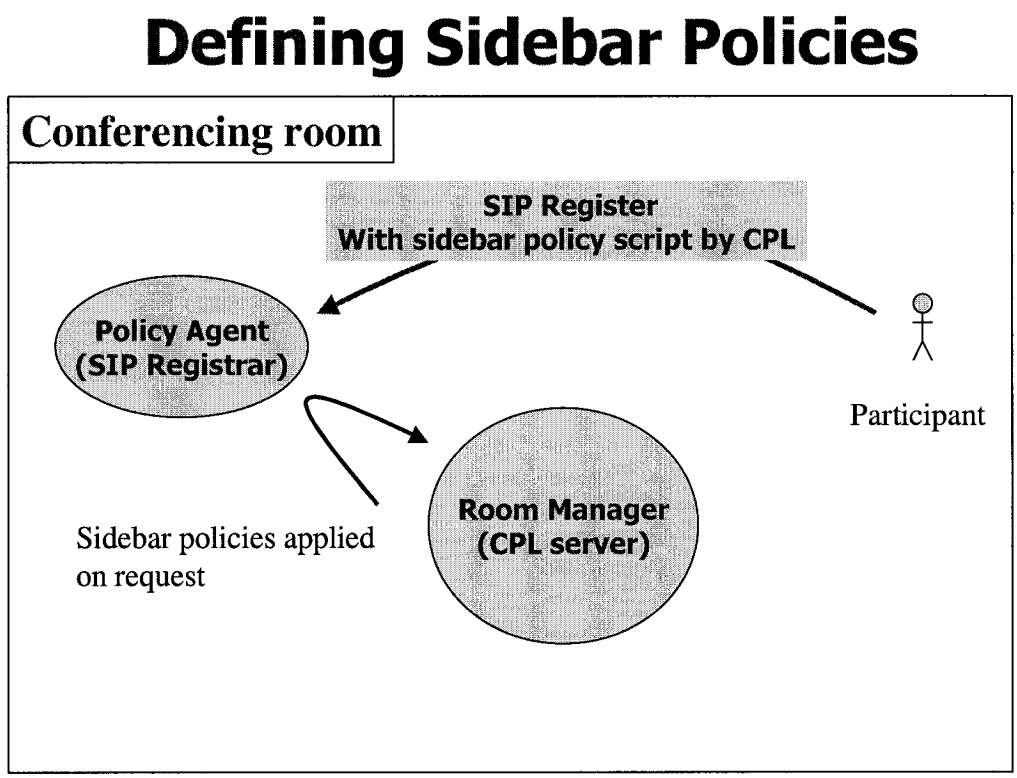


Figure 4.6 Defining sidebar policies

After defining sidebar policies, the particular user can initiate his/her sidebar. In figure 4.7, initiator 'A' originates a sidebar INVITE indicating that 'A' wants to initiate a sidebar with user 'B' with specified conferencing ID. This INVITE by default passes through RM. RM first authenticates initiator 'A's' actions, as only authenticated users could initiate a sidebar. If authentication is successful, RM will forward the SIP INVITE to user 'B'. After receiving a positive response from user 'B', the sidebar initiator 'A' creates the sidebar state and a sidebar conferencing between 'A' and 'B' starts. After the sidebar initiator 'A' has successfully created a sidebar state, he/she notifies RM about the new sidebar. RM subsequently sends a notification to qualified users about this new sidebar as shown below in figure 4.7.

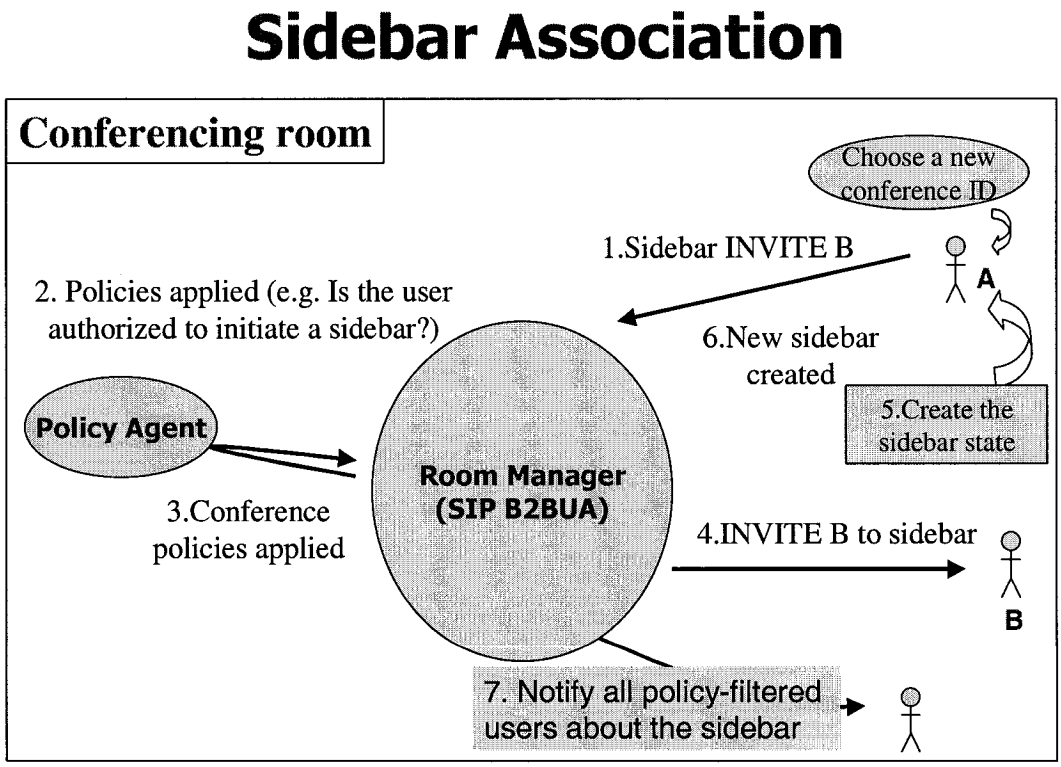


Figure 4.7 Initiating a sidebar

4.5.2 Joining a Sidebar

Procedures to join sidebars are different for different users.

- **Case 1**: A user may be notified about a particular existing sidebar in conferencing room based on the particular sidebar's policies. These users may choose to send a request to the particular sidebar initiator to join the sidebar.
- **Case 2**: Some other users may not be aware of this existing sidebar. These users may be invited by on-going sidebar members to join the sidebar.

These two cases are discussed separately as follows.

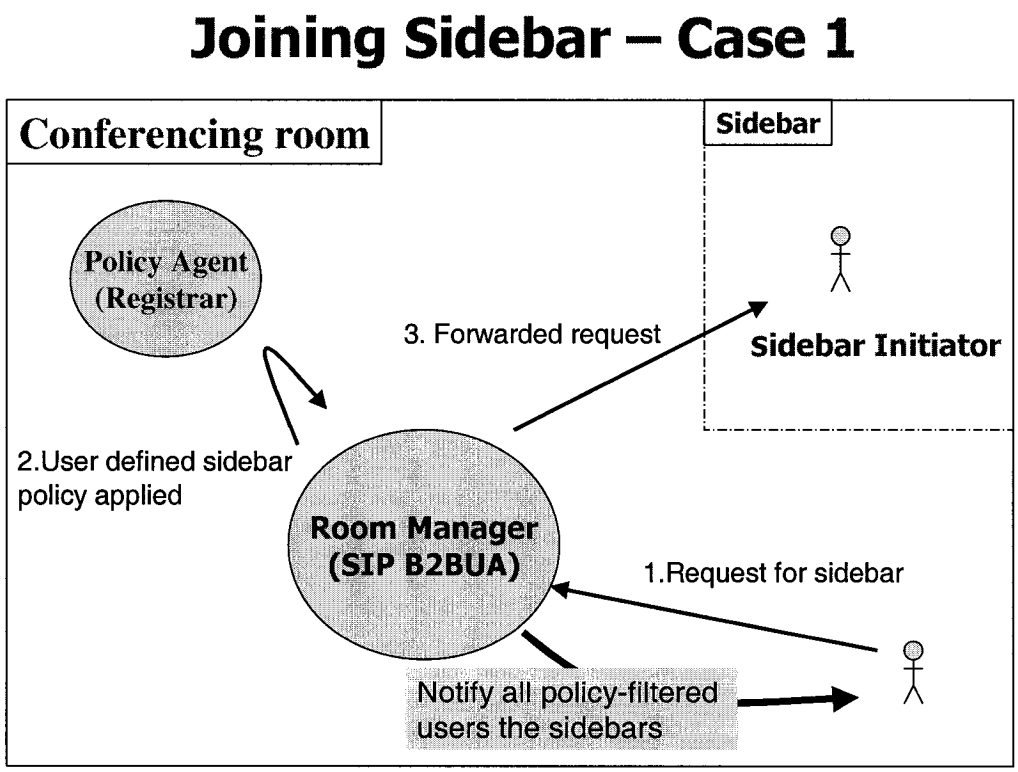


Figure 4.8 Joining a sidebar – case 1

Case 1: When a sidebar is created, by default no other participant in the conferencing room knows about this new sidebar. However, sidebar initiator may specify who has the right to know about his/her sidebar in the conferencing room. These users will get notifications about this new sidebar from RM and may choose to join the sidebar as shown above in figure 4.8.

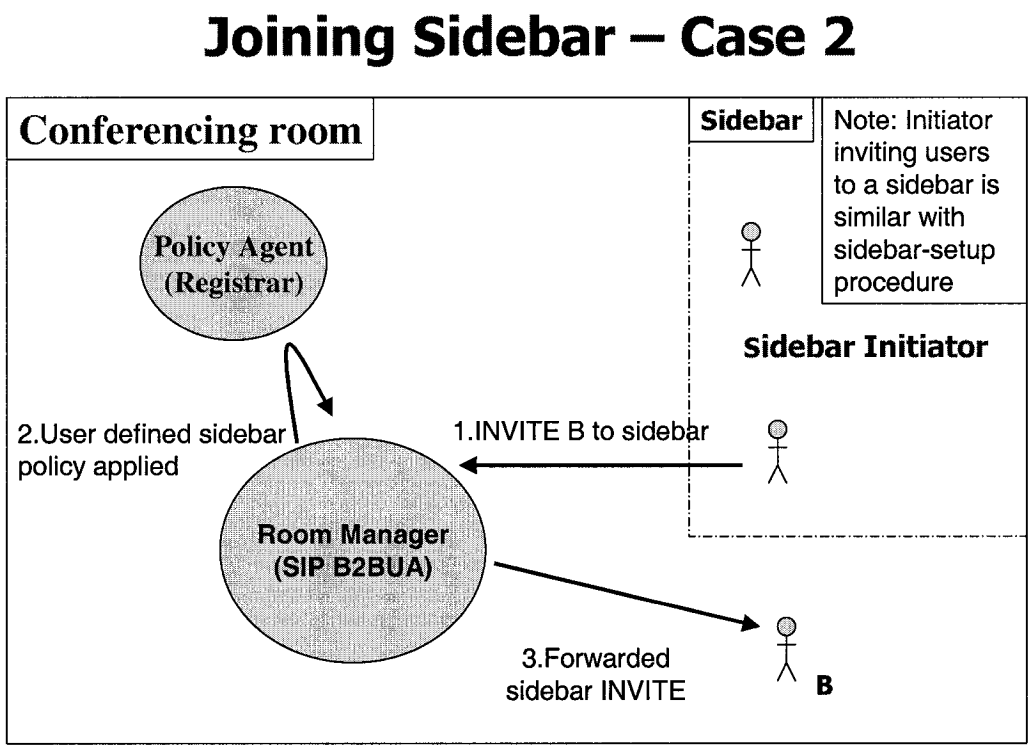


Figure 4.9 Joining a sidebar – case 2

Case 2: After a sidebar is created, either the sidebar initiator or any other sidebar participant can invite any other user to join this sidebar. Sidebar participants are allowed to invite conferencing room users based on on-going sidebar policies. The invitation process is carried out in the same way as in case 1. The case 2 is represented above in figure 4.9.

4.5.3 Leaving a Sidebar

Sidebar participants may choose to leave the sidebar without any restrictions by sending a SIP BYE to sidebar initiator. On the other hand, since the sidebar initiator has complete control of his/her sidebar, he/she can also 'kick' anyone out of the sidebar by sending a SIP BYE.

4.5.4 Terminating a Sidebar

When a sidebar is created, the sidebar initiator may have scheduled time for it. The sidebar state is terminated while the scheduled time arrives. Also, the sidebar initiator can terminate the sidebar state at any time, as he/she has full control over it.

Sidebar schedule time may not be indicated in certain cases and the sidebar initiator too may not terminate the sidebar state while leaving the room. In this case, when RM receives a notification regarding a sidebar initiator leaving the room, it will notify the sidebar initiator to terminate the on-going sidebar. Even the notification is not received by the particular sidebar initiator, RM will update sidebar storage and notify PA to clear all sidebar policies defined by this user.

4.6 Discovering Participant Identities

The ad-hoc conference state changes frequently as users join or leave the room, active participants change, media types change and so on. Conference participants may need to

be notified about these changes. According to Conference Event Package [43], a user needs to subscribe to the *conference state presentity*, so as to be notified of any change in the on-going conference. Since RM is responsible for maintaining ad-hoc conference states in the SIP-based ad-hoc conferencing system, conference participants should send requests to RM in order to subscribe. Consequently, whenever there is a conference change RM will generate notifications to subscribers.

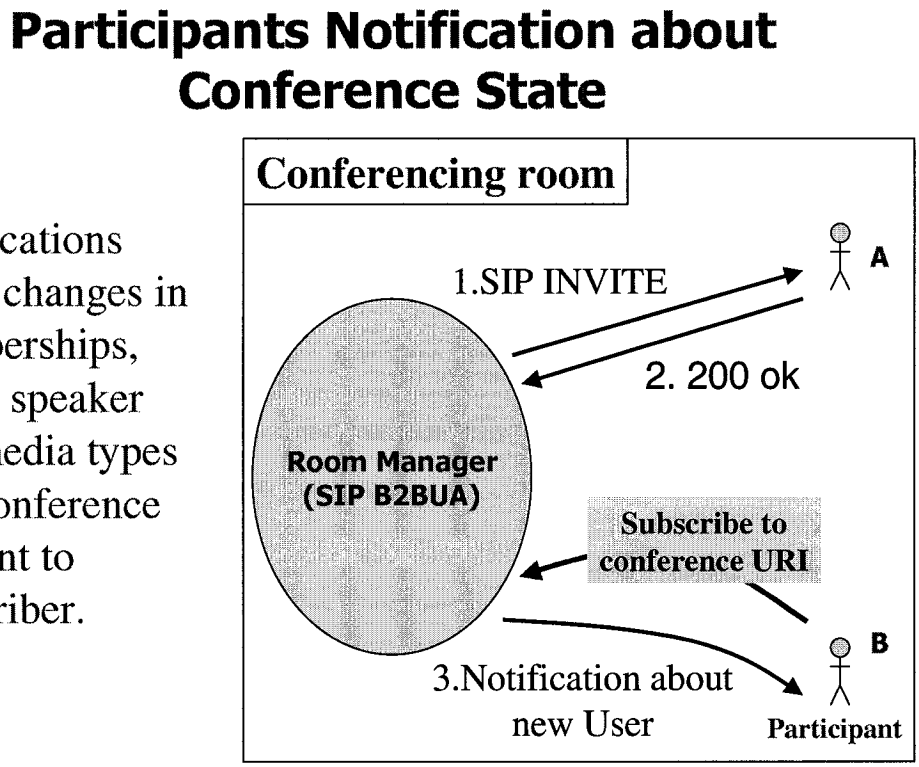


Figure 4.10 Conference participant notification

As shown above in figure 4.10 conference participant 'B' sends a SUBSCRIBE request to RM indicating that he/she wants to be notified of change in the conference with specified URI. When a new user 'A' enters the room and joins the on-going

conference, RM generates a notification about the same and immediately sends the generated notification to user 'B' (i.e., all participants with URI subscription).

Sidebar initiators are in charge of sidebars they create. Therefore, these initiators are responsible for handling subscriptions to the corresponding sidebars. Sidebar participants send subscription requests to corresponding sidebar initiators. When a sidebar state changes, the corresponding initiator sends notifications to corresponding subscribers about the same.

4.7 Summary

This chapter discussed SIP-based ad-hoc conferencing system in detail. The architecture, features and scenarios of SIP-based ad-hoc conferencing system are elaborated. The design includes ad-hoc conference set up and termination, source policy definition, sidebar initiation and termination, and notifications regarding conferencing changes.

Next chapter describes the implemented system prototype which discusses how the SIP-based ad-hoc conferencing system is set up and how the whole system works.

Chapter 5 System Implementation

This chapter discusses a prototype for the ad-hoc conferencing system discussed in this thesis. The tools and techniques for the implementation are introduced in detail. The chapter also presents selected snapshots and figures illustrating the discussed prototype.

The ad-hoc conferencing system implementation is agent-based. FIPA-OS v2.1.0 [47] agent platform is used to execute agents. Agents communicate by means of Agent Communication Language (ACL) [44]. SIP server and SIP UA are from Columbia University, sipd1.0 and sipc1.1 respectively. The ad-hoc conferencing system uses Robust Audio Tool (RAT4.2.14) and Videoconference Tool (VIC2.8) conferencing tools, which are "open-source audio and video conferencing and streaming application that allows users to participate in audio and video conferences over Internet" [45, 46]. Ways to efficiently use RAT and VIC conferencing tools are stated; which is a small but important issue and requires considerable amount of time. The overall implementation is done using Java 1.2.

5.1 FIPA-OS Agent Platform and Agent Communication Language

FIPA-OS (FIPA Open Source) [47, 48] agent platform supports multiple agents communicating with each other by means of Agent Communication Language (ACL)

[44]. The FIPA - Open Source platform was originated from Nortel Networks and was first released in August 1999. It is the first publicly available agent platform that supports the FIPA (Foundation for Intelligent Physical Agents) standard. It can interoperate with other FIPA compliant agent platforms and because of its openness it can easily be extended and evolved.

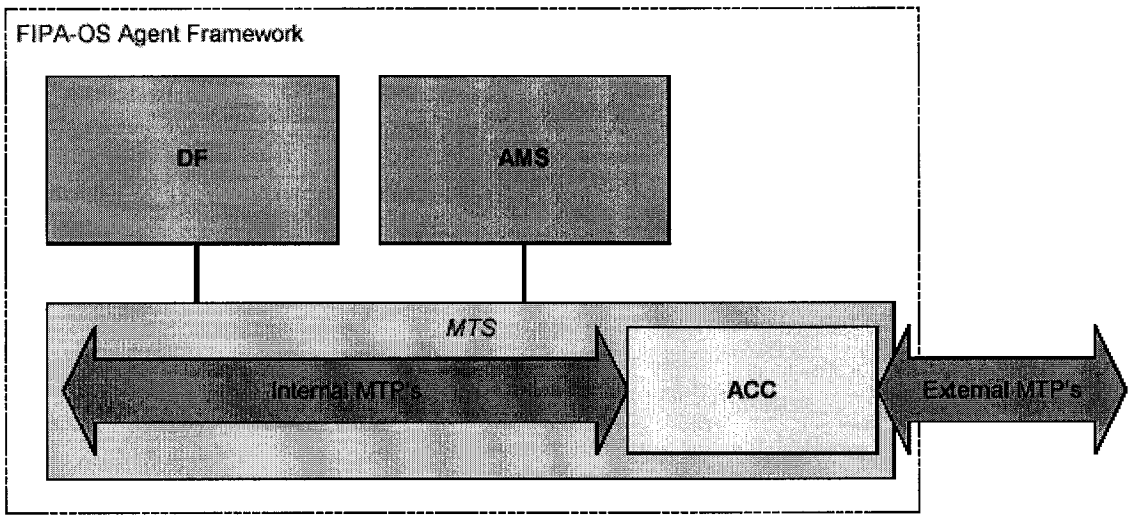


Figure 5.1 FIPA-OS Agent Platform [49]

As shown in above figure 5.1, the Agent platform consists of Agent Management System (AMS), Message Transport Service (MTS) and Directory Facilitator (DF) [49]. AMS and DF are agents. AMS is responsible for managing the agents running on the platform and DF provides other agents "yellow pages" services. MTS provides agents "message routing service". As one component of MTS, Agent Communication Channel (ACC) supports communications "channel" for agents on a particular platform.

Agents communicate and interoperate with other agents through single-platform communication or inter-platform communication. The FIPA Agent Communication Language (ACL) is the communication medium used for agent communication [44, 48].

The FIPA ACL message contains a set of elements that allows an agent to state its intentions. An ACL message is an expression whose arguments are standard in a universal way so that the information is clear to other compliant agents. The elements that form the ACL vocabulary and their purposes are listed below in figure 5.2.

Element	Category of Elements
performative	Type of communicative acts
sender	Participant in communication
receiver	Participant in communication
reply-to	Participant in communication
content	Content of message
language	Description of Content
encoding	Description of Content
ontology	Description of Content
protocol	Control of conversation
conversation-id	Control of conversation
reply-with	Control of conversation
in-reply-to	Control of conversation
reply-by	Control of conversation

Figure 5.2 FIPA ACL Message Elements [44]

The parameters needed for effective agent communication vary during communication. The only mandatory parameter is the *performative,* which phrases the type of agent communication that is in progress. *Sender*, *receiver* and *content* are other commonly used parameters. The *sender* and *receiver* parameters state senders' and

receivers' information including name, address and platform details. The *content* element contains the actual content that is being transferred.

All agents involved in the ad-hoc conferencing system run using FIPA-OS agent platform. They communicate using ACL to collaborate their activities by interacting with other agents in the conferencing system.

5.2 Tools Used and the Snapshots

5.2.1 SIP Call Snapshots

SIP is used in the ad-hoc conferencing system to negotiate media types and establish sessions for main conference participants. SIP server and SIP UA are from Columbia University [50], sipd1.0 and sipc1.1 respectively. A snapshot representing the SIP UA interface and SIP call initiation are shown below in figure 5.3.

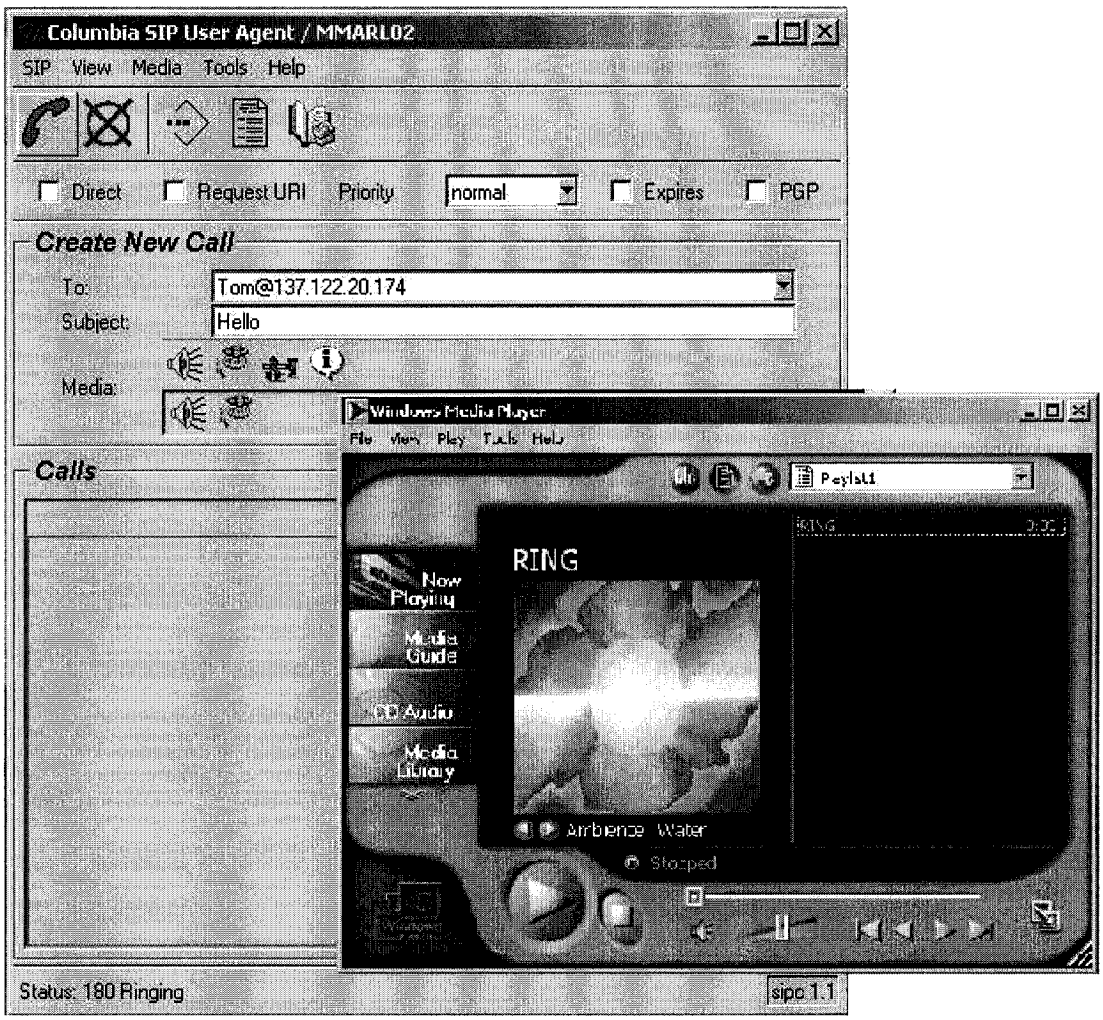


Figure 5.3 Initiating a SIP Call

Different fields in the SIP window are as follows. For every SIP INVITE, the callee is indicated in "To" field, session topic in "Subject" field, and chosen media type in "Media" field. In the above shown figure 5.3, the owner of displayed SIP UA sends a SIP INVITE to user "Tom" registered at SIP server "137.122.20.174". The suggested media types are audio and video. The callee's ("Tom") SIP UA has received this INVITE and sends back "180 Ringing" response to the caller. This response is shown as "Status" in status bar at bottom-left corner of the window indicating that this UA is waiting for

callee's further response. As a 'response' to "180 Ringing", the SIP UA's of both caller and callee automatically start media player to 'play' call ringing sound.

At the callee's end the UA also pops up a window that prompts the user about the incoming call. The window also displays incoming call's 'source id' and 'subject'. The user can then accept or reject the call by clicking the appropriate button in the window prompt. This is shown below in figure 5.4.

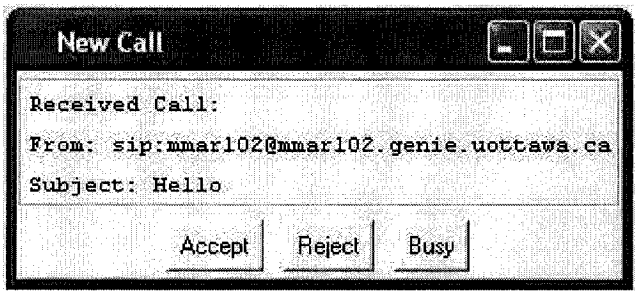


Figure 5.4 Receiving a SIP INVITE

If the callee clicks "Accept", callee's SIP UA sends back "200 OK" to caller's UA, which will again be displayed in status bar. Thereupon, a session is established between these two users as shown below in figure 5.5. The UA interface in the left side of the figure shows the caller in an established session and the interface in the right side of the figure shows the callee in an established session. In both interfaces, users may click "Bye" button to send a "BYE" at any time. This will terminate the on-going dialogue.

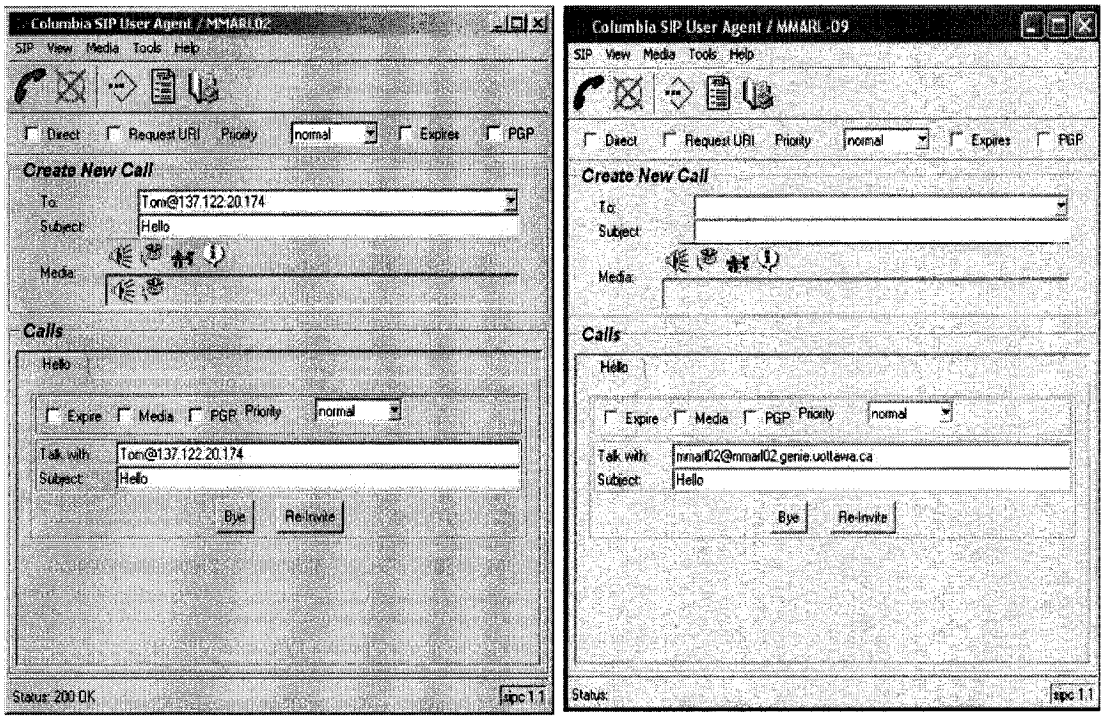


Figure 5.5 An Established SIP Session

5.2.2 Audio Conferencing Tool – RAT

RAT (Robust-Audio Tool) is a tool developed for audio conferencing over the Internet. RAT is used for multiparty audio conferencing. It can be started from command line as follows [51],

Prompt> rat [options] < IP address/port>

For multicasting, IP addresses must be in the range 224.2.0.0 - 224.2.255.255 (except while using admin scope). The port number must be an even number and at least 1024. The IP address and port number indicates the address wherein multicast

conferencing could be started. All participants must start RAT at the same IP address and port number so as to take part in the same multicast conferencing.

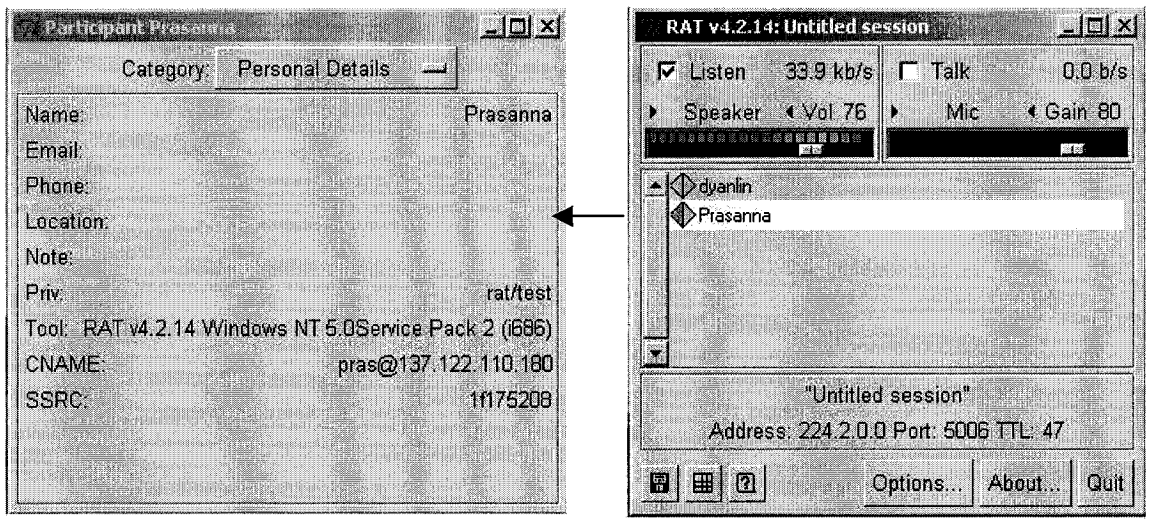


Figure 5.6 Main RAT Window

As shown above in figure 5.6, RAT has a main window to display all conference participants (right figure). The current speaker will be highlighted. By clicking on any participant, a new window is displayed which details user profile such as name, e-mail, and transmission status. The checkbox "Talk" on top right corner of the main window should be checked in order for microphone to start audio transmission. In this example, there are two users in an on-going conference and the highlighted user "Prasanna" is speaking.

5.2.3 Video Conferencing Tool – VIC

VIC is developed by Network Research Group at Lawrence Berkeley National Laboratory and University of California, Berkeley. Video Conferencing Tool is used for video conferencing over the Internet. VIC is used for multiparty video conferencing. It can also be started from command line as follows [52],

Prompt> vic [options] < IP address/port>

For multicasting, IP addresses must be in the range 224.2.0.0 - 224.2.255.255 (except while using admin scope). The port number must be an even number and at least 5002. The IP address and port number are needed to start a multicast conference. All participants must start VIC at the same IP address and port number so as to take part in the same conference.

As shown below in figure 5.7, VIC has a main window to stream all conference participants' videos (top left figure). Besides video streaming, participants' name and email are also displayed. A button labeled 'info' is located under user information. Clicking 'info' button will lead to a window displaying a user's detail profile, including RTP status. Clicking on any participant's 'thumb' image will lead to a new window displaying the corresponding enlarged picture. The 'Menu' button at the bottom of the main window is provided to open a 'Menu' Window (right figure). By checking 'transmit' displayed on top of 'Menu' window, the corresponding users' image will be transmitted to other conference participants. By clicking 'Release' in the same 'Menu'

window, the on-going transmission can be halted. At this time, other participant's window display streaming video will 'freeze'.

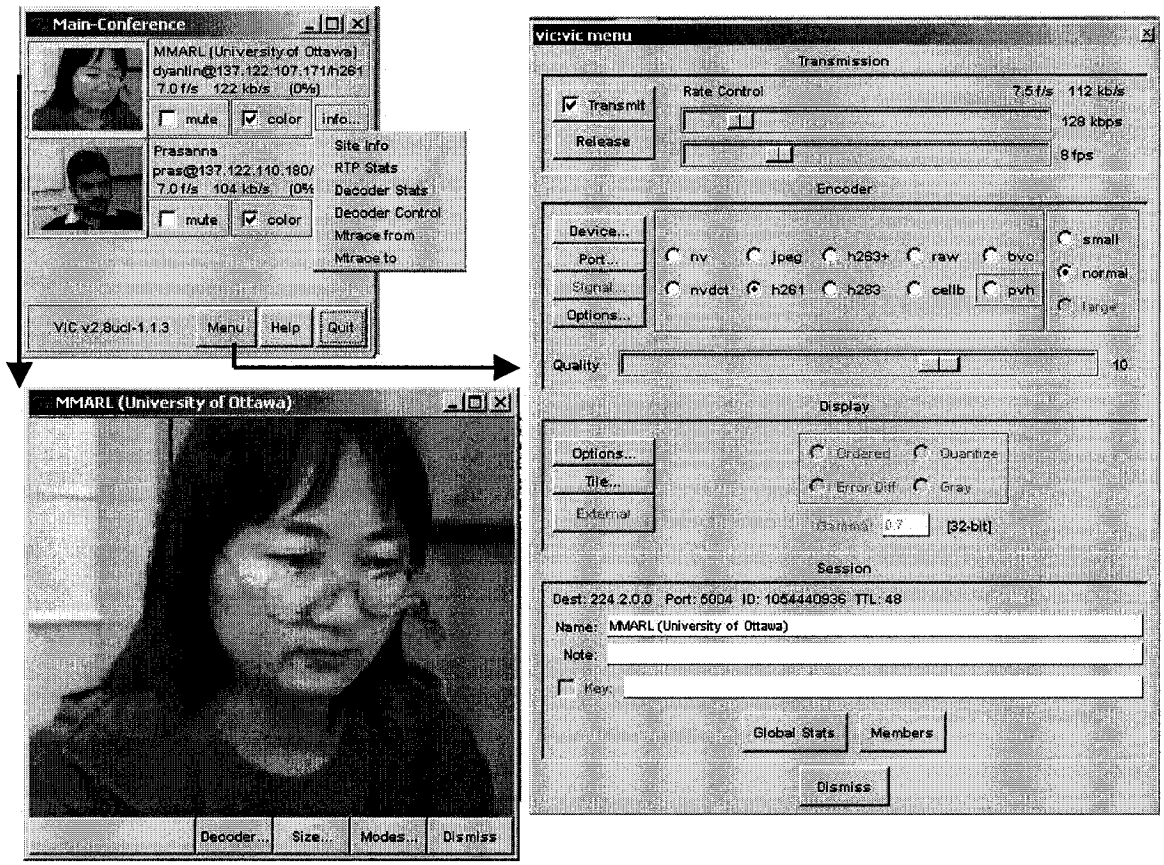


Figure 5.7 Main VIC Window

5.3 Prototype Description

5.3.1 The Mobile Ad Hoc Communication Project

This SIP-based ad hoc conferencing system provides conferencing and sidebar association features as a part of ad-hoc communication project. The central idea of mobile ad-hoc communication project is to bring various types of users and services

together into a network where they can collaborate with one another and share services in the network.

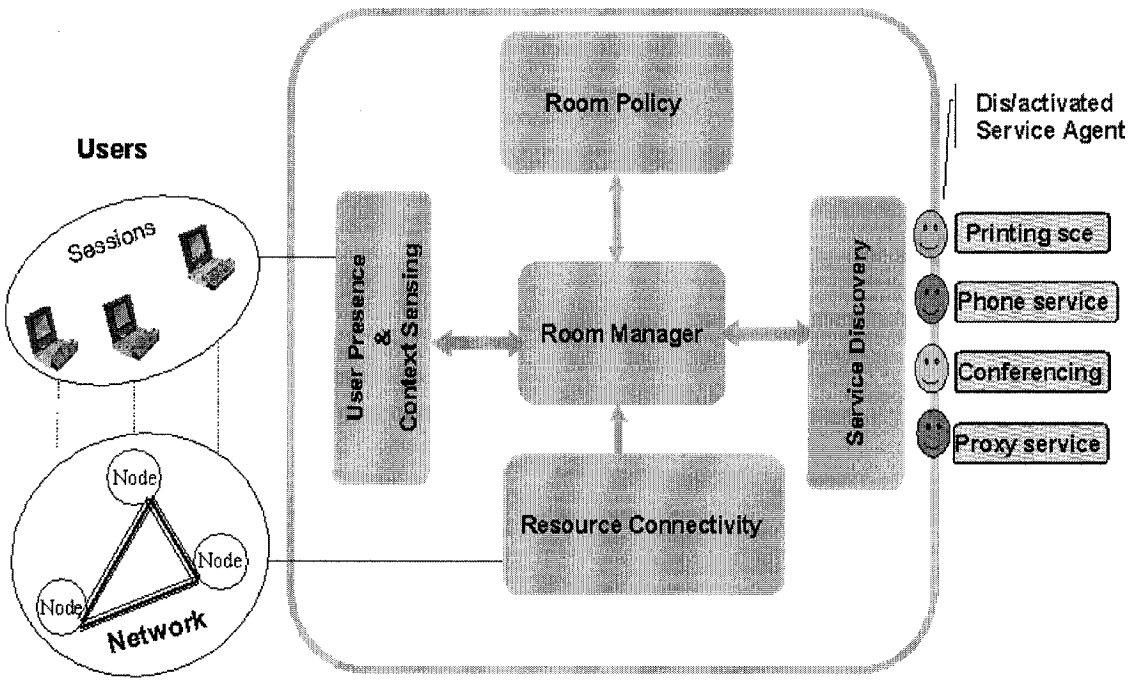


Figure 5.8 Conference management as a part of Ad hoc Communication Project

The above shown figure 5.8 represents the ad-hoc communication project with RM as the central controller. The main purpose of this project is to enable collaboration between different entities, which are users and services. The users and services join and leave the room in a dynamic fashion. The users may or may not carry personal devices such as laptops or PDAs. The presence of these entities is identified spontaneously and appropriate tools for collaborating with environment are supplied to client devices in case they do not possess the required tools. The underlying network can be wired or wireless depending upon the device that are connecting to it. The context agent (CTA) provides context sensing and user presence information to RM. The CTA encompasses all the sensors in the environment that provide context information about entities in the room.

The CTA identifies resources and their attributes uniquely and supplies this information to service discovery and resource connectivity modules through RM.

RM couples the context information from CTA with service discovery mechanisms along with policies of participating entities in the room to enable collaboration among people and services. Every privileged user is made aware of the services and other users in the room after passing policy tests. The users are allowed to share their own services and use the available authorized services when they need and an exclusive service discovery module takes are of it. The service discovery provides the suitable service matching the request from the client in the same room or from other similar rooms. Once the target devices are identified by their capabilities, a communication session is formed between them and monitored constantly. This session is managed by using the context information of the room and that of the participating entities. The resource connectivity module is responsible for creating and managing the communication sessions between entities. It provides session management for the user – service or user - user interaction.

Among all the services provided to users, the conferencing service is managed and controlled by this SIP-based ad-hoc conferencing system. Users dynamically register with a conference association by entering the room and therefore enjoy various features provided by this conferencing system.

5.3.2 Implementation Details

In the prototype, the workload of service management is distributed over RM and different service agents.

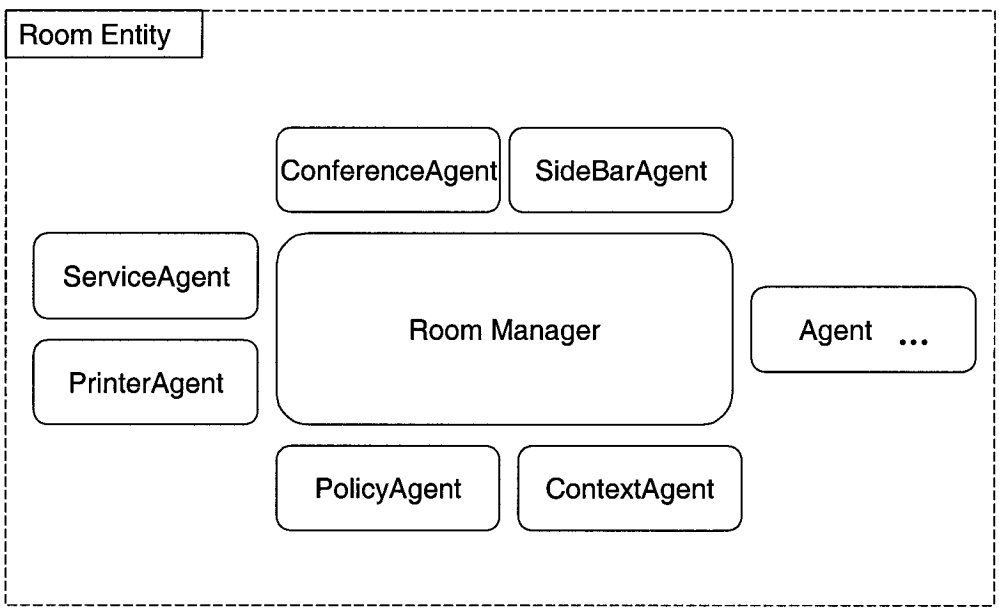


Figure 5.9 Agents in the Ad Hoc Communication Project

RM is responsible for coordinating and monitoring overall conferencing room tasks. Each service agent concentrates on managing one service. The distributed arrangement provides more flexibility and convenience to implementation and improves the performance of the system. For example, the ContextAgent monitors context change, the PrinterAgent provides printer service and PolicyAgent takes care of constraints.

ConferenceAgent (CA) and **SidebarAgent (SAgent)** provide conferencing and sidebar association features to users in the ad hoc network.

User Personal Agent - UPA

User personal agent runs on a user's terminal and works on behalf of the particular user. When users enter a conferencing room to take part in conferencing, the only software that is mandatory is FIPA-OS platform. A new agent named MMA is inserted in FIPA-OS platform and will start along with FIPA-OS. The purpose of MMA is to notify RM about a new host in the network. RM will then send a cloned UPA to the corresponding user. MMA receives the required information and activates the UPA. Thereupon, for every user in the room, an UPA works on behalf of him/her to interact with other agents for processing users' requests.

UPA provides a GUI to every user for displaying other users and services in the network with which the user can collaborate. The service and user lists are displayed after performing policy checks for users and services. If the user wants to use a particular service, he/she selects the required service from the list and clicks 'submit' (Examples of GUI are shown in figure 5.18 and 5.22). The request goes to the appropriate service agent and this agent migrates to corresponding user terminal to perform further operations.

Room Manager - RM

RM is an agent working on top of SIP UA. When it receives a SIP request, it always checks the maintained policy for further actions and decisions. It then creates a new appropriate SIP response and sends it out. At the same time, RM is an agent, so it is capable of handling requests in both SIP and ACL format.

Users in a conferencing room need software like SIP, RAT, and VIC to take part in a conference and communicate with other people. When one user enters the room, RM will check if the user has necessary tools to take part in a conference. RM sends a software package to 'entered' user in case the user doesn't own any of the required software. The interactions rely on the communication between user's UPA and RM. If RM does send a software package to an 'incoming' user, UPA is responsible to receive the software package and start SIP UA immediately.

Another important operation of RM is registering to the SIP server on behalf of users. After a user enters the conferencing room and joins the conference, RM sends out a new SIP REGISTER request with higher priority to overwrite the original SIP register made by user's SIP UA. This allows RM to intercept all INIVTEs to the SIP UA in the future. In this way, RM takes overall control over any established session involving the SIP UA. If needed, RM may reject incoming INVITEs to a SIP UA in the room, so as to avoid conferencing interruption.

5.3.3 Conferencing Prototype

The following represents an overview of the SIP-based ad hoc conferencing management framework. Various modules interoperate together to form the conferencing management system under discussion and the individual module in the prototype is described in detail in this section.

Modular view of the conference model

The conferencing management framework consists of Agent Deployment Block, Data Storage, Query Handler and Application Interfaces as shown below in figure 5.10.

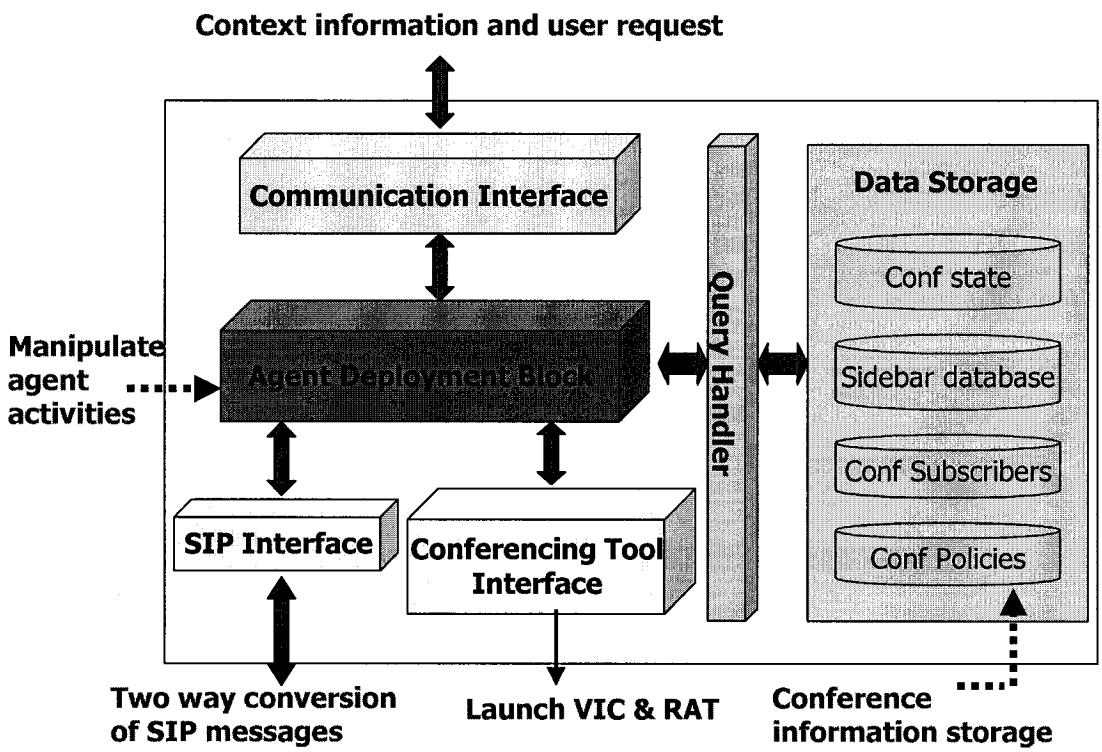


Figure 5.10 Modular view of the conference model

Agent Deployment Block is the core of the conferencing model and controls all operations performed by the agents. *Data storage Block* acts as the information base containing conference data in the structure and is responsible for managing stored conferencing states and maintaining the integrity of data. System handles the requests and replies to the conference data storage through *Query Handler*. The *agent deployment block* and the *data storage block* can be coupled together as agents manage conferencing activities and control the conference data storage and retrieval. *Application Interfaces* deals with communications with external applications that are either agent based or non-agent based. It handles information coming from lower context module, provides APIs for user interaction and integrates SIP and conference tools into the conferencing system.

Implementation prototype of the conference model

All communications between internal and external modules are purely agent-based. Figure 5.11 illustrates the usability of agents in this SIP-based ad hoc conferencing management system architecture. The agents run on top of FIPA-OS agent platform, provide interfaces for user interactions and communicate with other agents.

The CA consists of two agents, namely the Conference Agent and SIPWrapper Agent. The Conference Agent receives context change information from the lower context module and controls all the conferencing activities in the main ad hoc conference. The SIP wrapper agent integrates SIP with the conferencing system and converts SIP messages in two ways.

The Sidebar, SidebarInitiate and SidebarReceive agents together process sidebar requests from users. When there is a request for the sidebar service, these three agents cooperate with each other to perform the task.

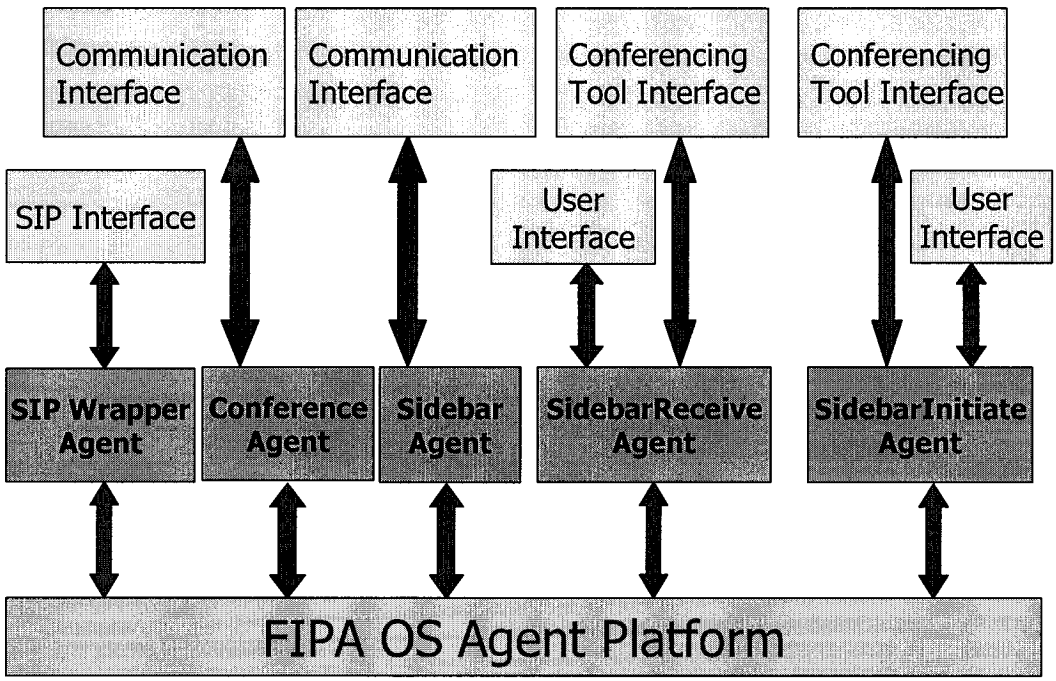


Figure 5.11 Implementation prototype of the conference model

Agent communication

Figure 5.12 below shows how the agent communications are carried out under different circustances. All information exchanged between agents is in the form of ACL messages.

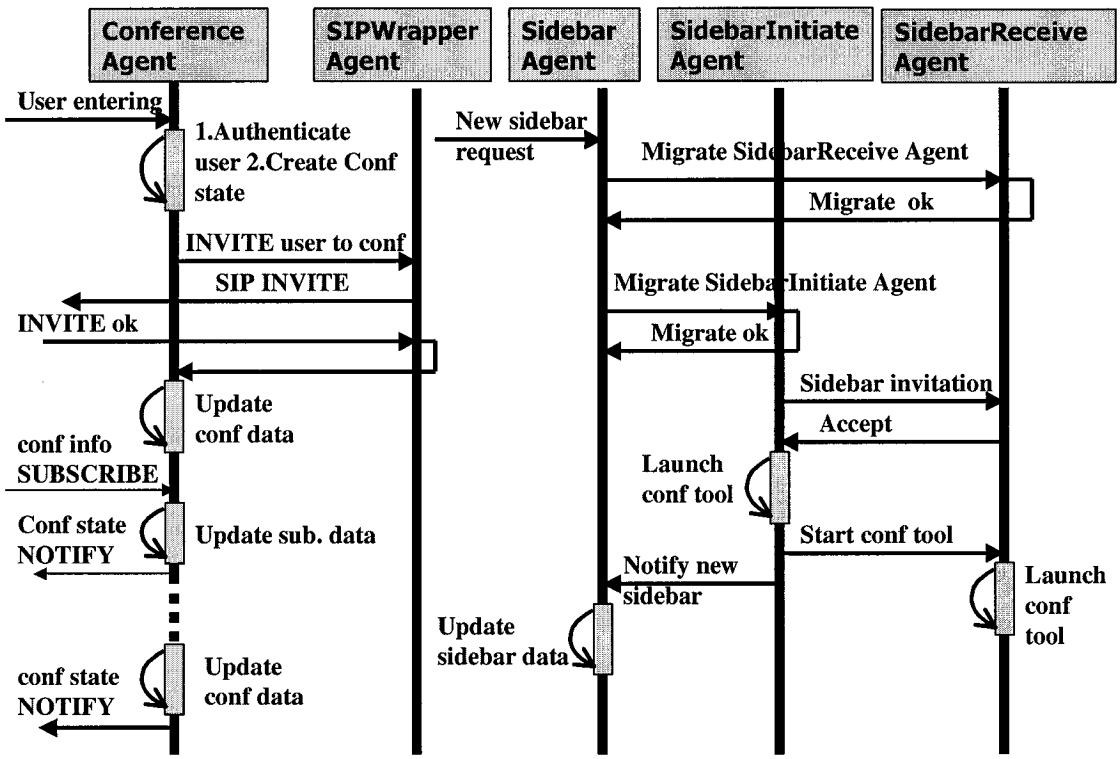


Figure 5.12 Communication between agents in the prototype

When a user is entering the room, the context change information is passed to the Conference Agent. The Conference Agent authenticates users and creates a conference state. The SIPWrapper Agent then invites the user to the ad hoc conference by using SIP and sends back the result to the Conference Agent for updating the conference state. When there is a SUBSCRIBE request regarding the changes in the main ad hoc conference state, the Conference Agent receives the request and sends a corresponding notification to the subscriber immediately. The Conference Agent also refreshes notifications to the subscriber whenever the conference state changes in the future.

When a sidebar request is received, the Sidebar Agent processes the received request. The Sidebar Agent receives the request and migrates the SidebarInitiate and SidebarReceive agents to the sidebar initiator and the callee. The migrated agents running on users' terminals interact with each other to establish the sidebar association and load conference tools for these two users' communication. The Sidebar Agent is notified about the new sidebar in order to update the sidebar data.

A sidebar request is a move request that necessitates sidebar agents to be migrated to the sidebar initiator and the callee separately. In case of a move request, the agent requests the mobility management agent to move its object to the destination. Agent mobility is one of the key issues that was encountered while trying to model the conferencing system. As the FIPA OS agent platform does not feature agent mobility, agent cloning using Apache technology is the technique used in the system for migrating agents.

5.4 Snapshots of the Implementation

5.4.1 The Main Ad Hoc Conference

As we discussed before, when a user is entering conference room, RM notifies CA about the new user. Using SIP, CA invites this user to join the main conference.

5.4.1.1 Main Conference Setup

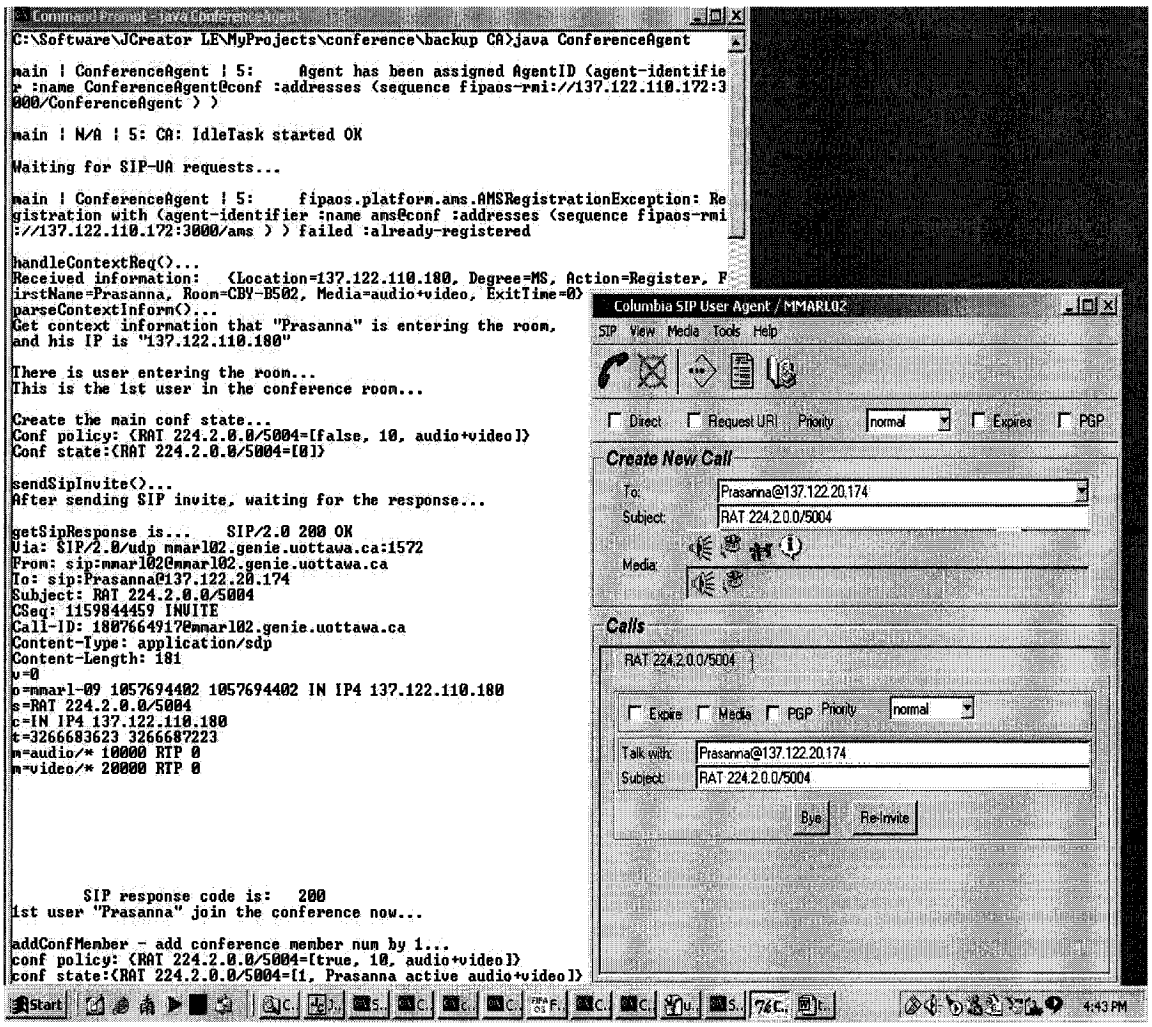


Figure 5.13 Inviting the 1st user to the main conference

Above shown in figure 5.13 represents main conferencing processing in the CA. When CA gets context notification of the first user entering the room, it creates the conference state and checks obligatory conference policies. The implemented conference in the thesis limits total conference participants to 10 and indicates that media types such as audio and video are permitted. After applying matching policies, CA sends a SIP INVITE

to user "Prasanna". In the INVITE, CA informs the user joining the conference with address IP 224.2.0.0 and port number 5004. If user 'Prasanna' accepts the invitation, he joins the conference and the conference state is simultaneously updated by the CA. In figure 5.14 shown above, the left window represents the work of CA and the right window represents the working interface of SIP UA on the CA.

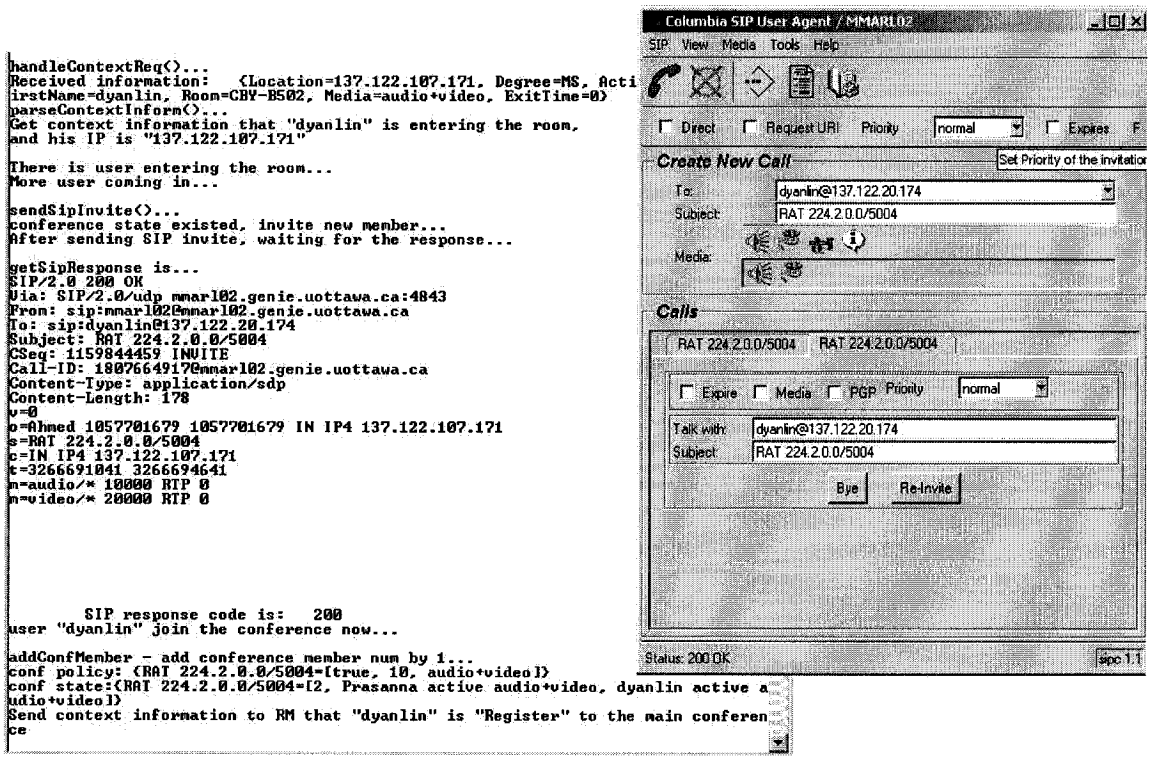


Figure 5.14 More users joining the main conference

CA establishes new SIP sessions with other new users entering the room and adds them to the conference member list by updating the conference state as showed above in figure 5.14 as well.

5.4.1.2 Communication in the Main Conference

Authorized users after automatically joining the main ad-hoc conference, receive SIP INVITEs from the CA. Users' SIP UA automatically starts up audio and video conferencing tools (RAT and VIC). As shown in below figure 5.15, a conference participant' window displays several separate windows. SIP window displays the conference subject and to whom the session is established with. RAT window displays the list of participants and the 'speaker' is highlighted. VIC window transmits participants' images.

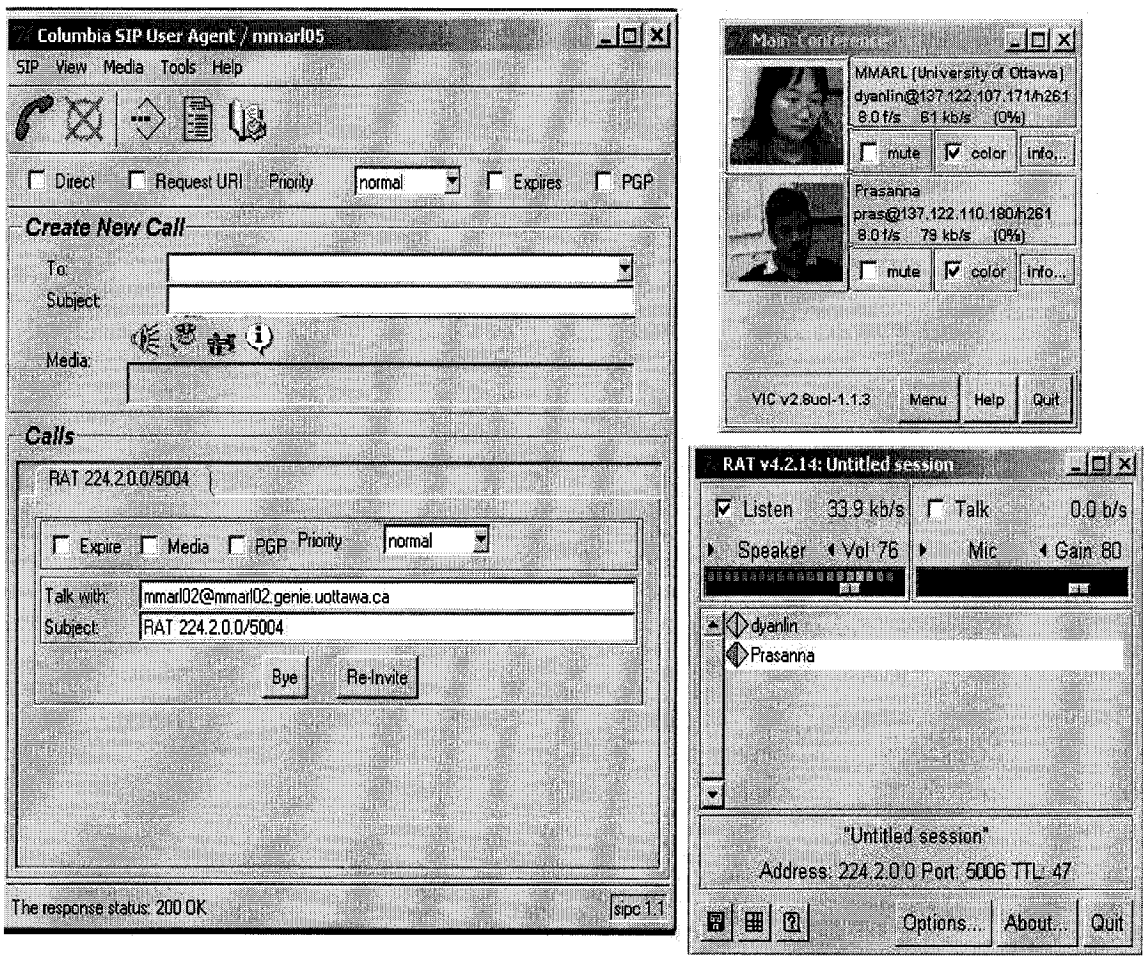
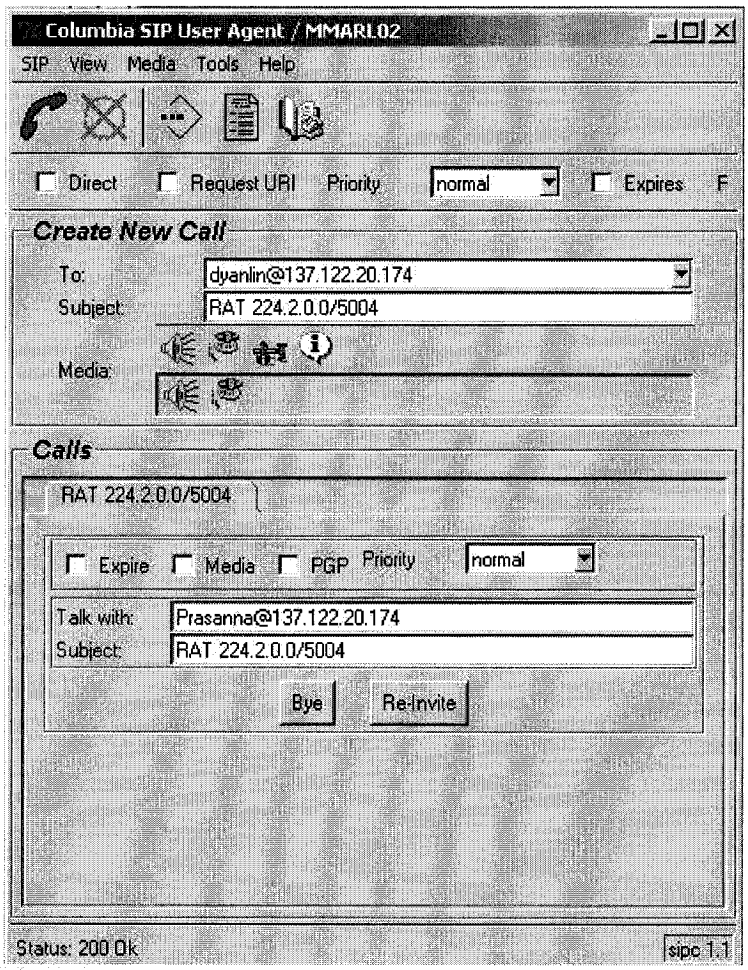


Figure 5.15 User's communication in the Main Conference

5.4.1.3 Leaving the Main Conference



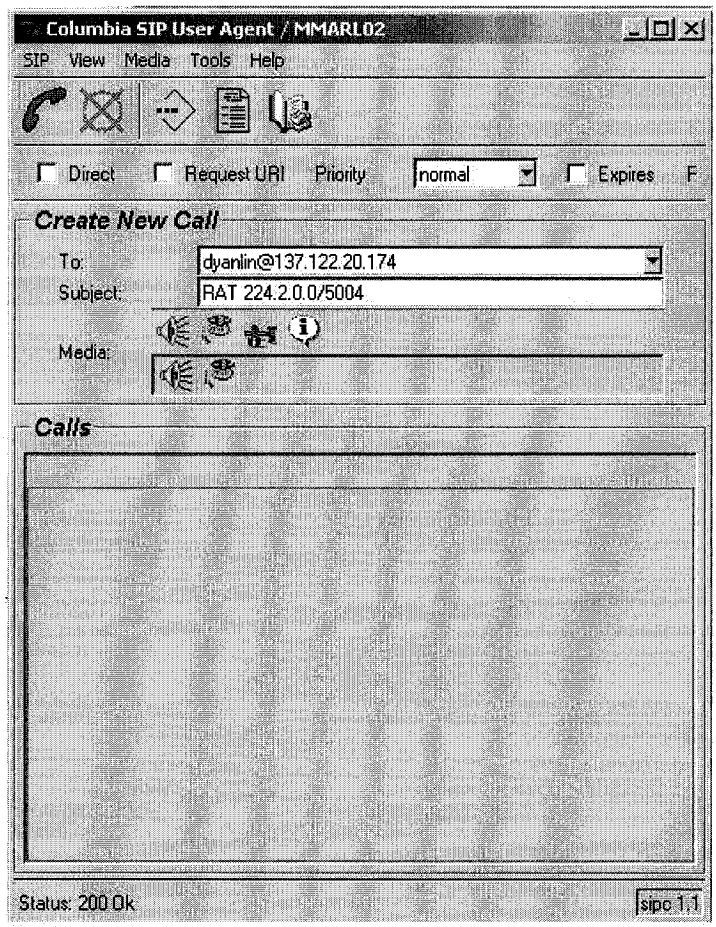

```
handleContextReq()...
Received information:   <Location=137.122.107.171, Degree=MS, Action=leave, Firs
tName=dyanlin, Room=CBY-B502, Media=audio+video, ExitTime=0>
parseContextInform()...
Get context information that "dyanlin" is leaving the room
Send SIP BYE to user "dyanlin"

deleteConfMember - decrease conference member num by 1...
conf policy: <RAT 224.2.0.0/5004=[true, 10, audio+video]>
conf state:<RAT 224.2.0.0/5004=[1, Prasanna active audio+video]>
There is user leaving the room...
```

Figure 5.16 Leaving the conferencing room

When a user is leaving, RM sends a SIP BYE to the user to remove him/her from the on-going conference. Figure 5.16 above shows the output of the CA. When user "dyanlin" leaves the room, the session between her and the CA comes to an end. It can be seen from

figure 5.16, only the call with "Prasanna" remains. CA also updates conference member list.

If there is no user currently presents in the room, all existing SIP sessions will be closed and CA terminates the conference state and clears all related data. CA output in figure 5.17 shown below represents the SIP interface with 'no' session.




```
handleContextReq()...
Received information:   <Location=137.122.110.180, Degree=MS, Action=leave, Firs
tName=Prasanna, Room=CBY-B502, Media=audio+video, ExitTime=0>
parseContextInform()...
Get context information that "Prasanna" is leaving the room
Send SIP BYE to user "Prasanna"

deleteConfMember - decrease conference member num by 1...
No user in the conference now...
The last user leaving the room...
Clear conference states...
```

Figure 5.17 The last user leaving the conferencing room

5.4.2 Sidebar Association

As discussed before in section 5.3, authorized users in the conferencing room have UPAs with GUIs working on behalf of them. Using these GUIs, users can send out requests to access services provided in the conferencing room. When one user wants to initiate a sidebar association with another, he/she selects sidebar members from a user list and chooses "SideBar" service from a service list, as shown below in figure 5.18. After this request is submitted, it goes to the appropriate agent, i.e., the SAgent for further processing.

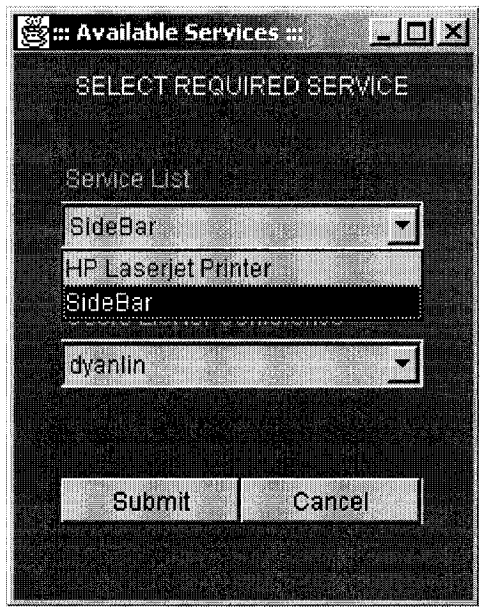


Figure 5.18 Initiating a Sidebar request

When a sidebar request is received, SAgent first checks whether the caller and the callee have sidebar agents to process a sidebar conference. If this is not the case, SAgent will send sidebar agents, namely SidebarInitiate Agent and SidebarReceive Agent, to

both parties in order to start a sidebar communication as shown below in figures 5.19 and 5.20.

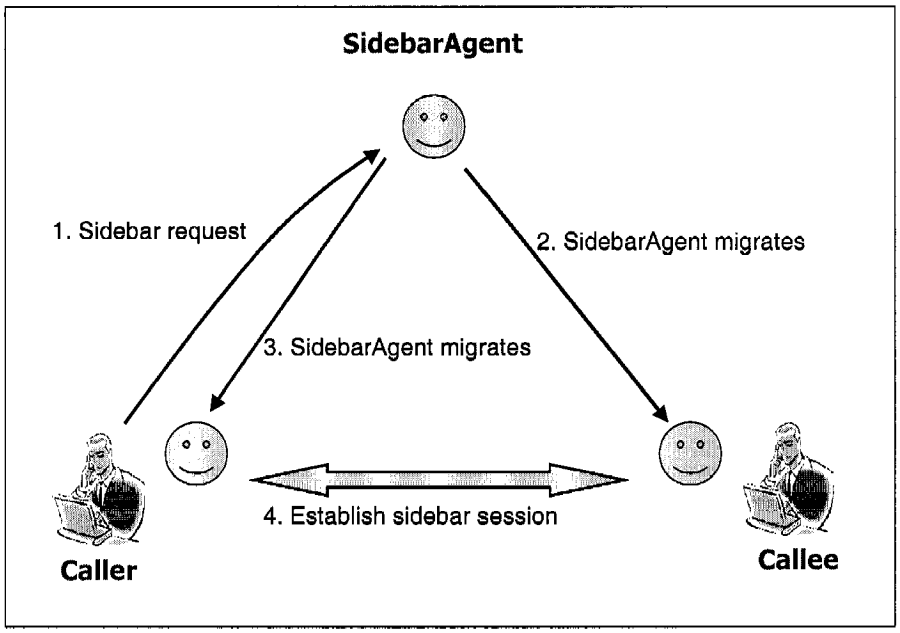


Figure 5.19 Migration of the Sidebar Agent

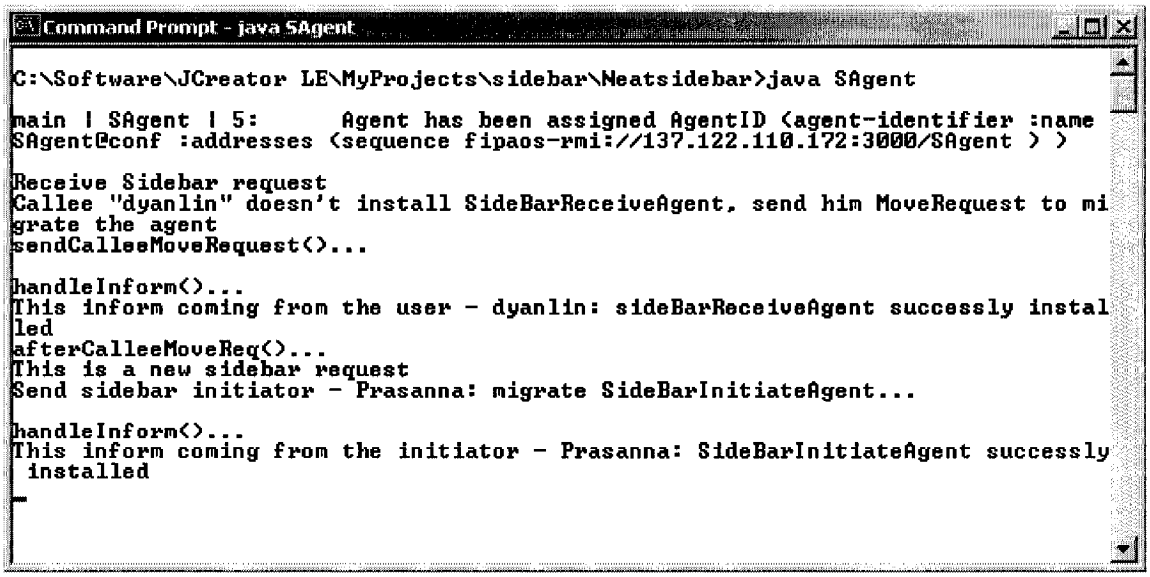


Figure 5.20 Snapshot representing the Sidebar Agent migration

After copies of Sidebar Agents migrates to platforms of both the sidebar initiator and the callee, the Sidebar Agents on users' platforms 'take over' the sidebar processing and collaborate with each other to establish a sidebar session between these two users as shown below in figure 5.21.

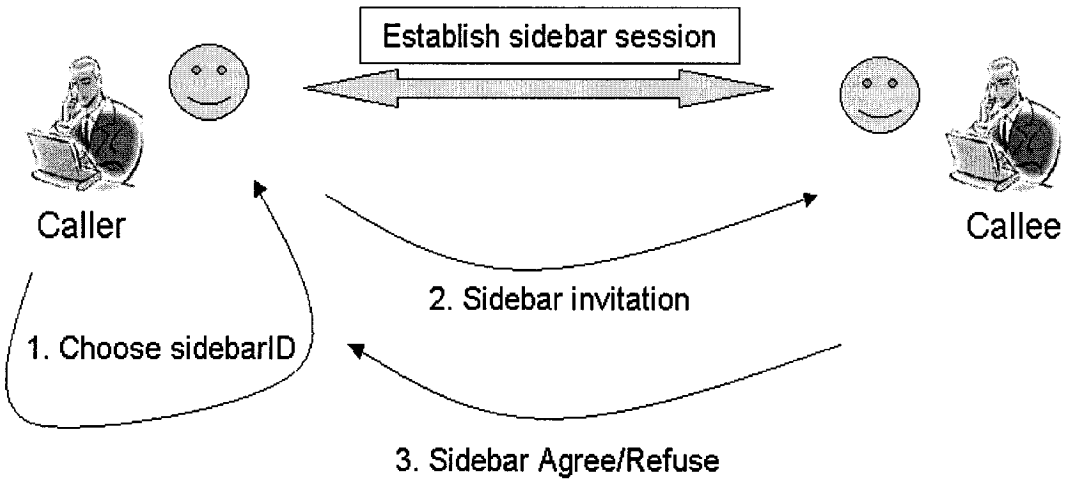


Figure 5.21 Establishing a sidebar session

The Sidebar Agent of the initiator, i.e. SidebarInitiate Agent, chooses a conference ID for the sidebar and specifies multicast address and port number for the sidebar. It creates the sidebar state and launches conferencing tools waiting for other users to join. The sidebar initiator then sends a sidebar invitation to the callee that includes the conferencing address. If invitation is accepted, Sidebar Agent of the Callee (SidebarReceive Agent) launches 'selected' conferencing tool for sidebar communication.

The sidebar initiator then notifies RM of the new sidebar through SAgent. RM distributes sidebar information to other conference participants. The policy-filtered users

may also request to join the on-going sidebar by sending a request to SAgent, which in turn passes these requests to the sidebar initiator. If the sidebar initiator approves received requests, his/her Sidebar Agent will send the sidebar association information to corresponding requested senders. Thereafter, appropriate users can launch RAT at the indicated sidebar multicast address to communicate with other sidebar members. A Sidebar member may also invite other users to join the sidebar. If the invitation is accepted, the invited party launches conference tool RAT and joins the corresponding sidebar.

5.4.3 Discovering Participant Identities

Conference participants send a SUBSCRIBE request to room entity to make a subscription to the main ad-hoc conference.

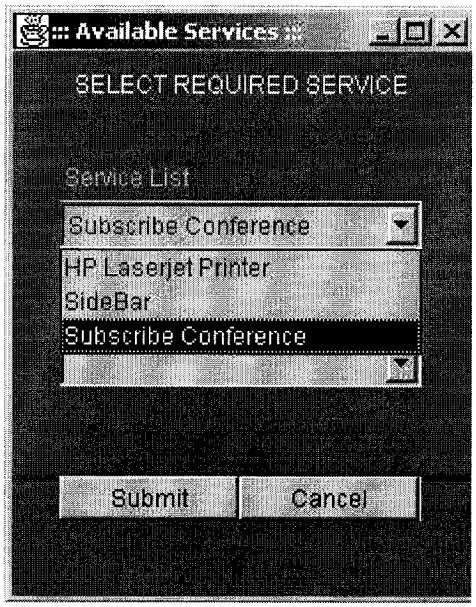


Figure 5.22 Initiating a Subscribe request

As shown above in figure 5.22, a participant can choose "Subscribe Conference" from the service list represented in the GUI. By doing so, the user initiates a SUBSCRIBE request to the room entity. The request is processed by the CA. CA associates the subscribe request with the main conference ID. Whenever a new user joins or leaves the main conference, the room entity notifies main conference's subscribers about the same by sending an updated conference state description. The main conference subscription details are schematically described below in figure 5.23.

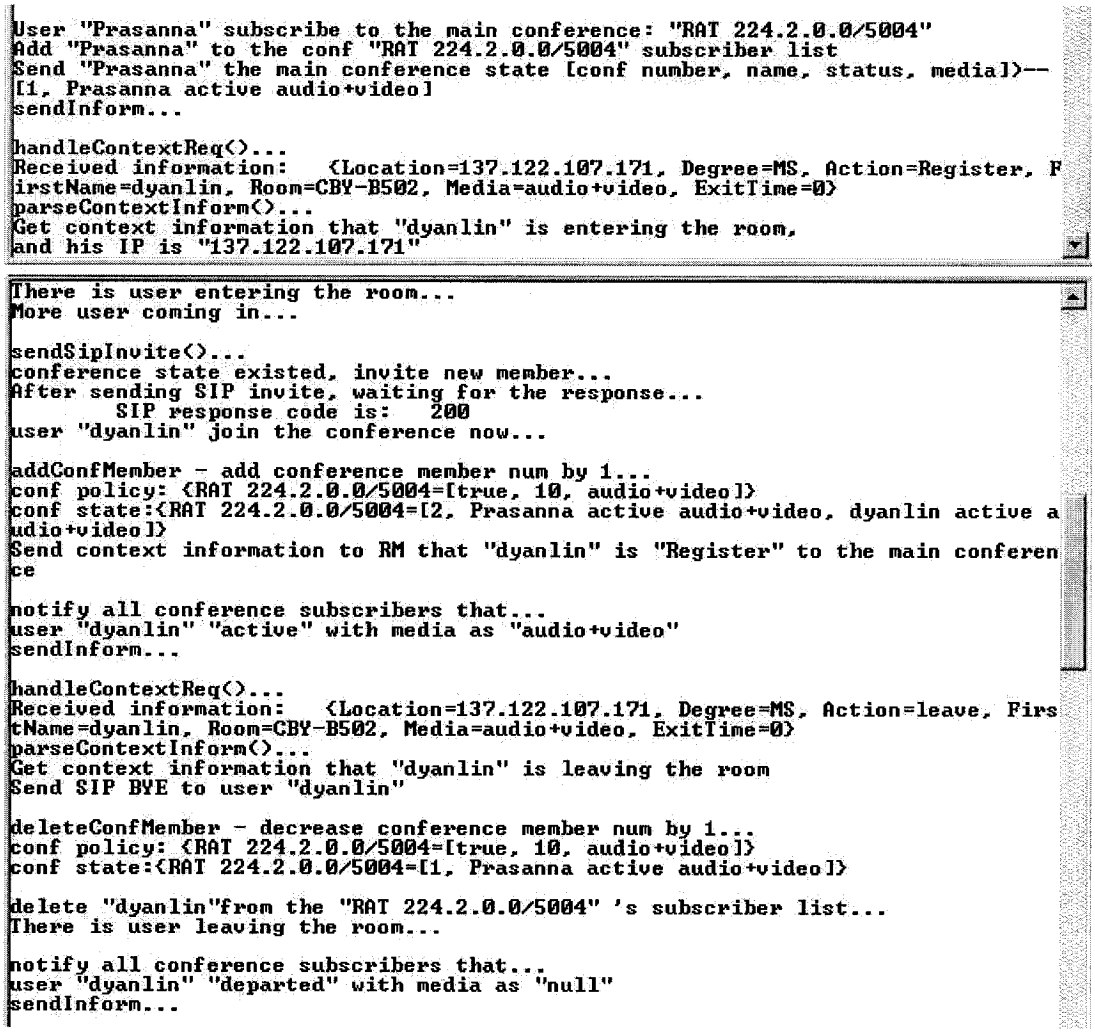


Figure 5.23 Snapshot representing main conference subscription

5.4.4 Conference Synchronization

Conference users face synchronization issues while using communication tools (RAT and VIC). Sidebar agents and PUAs should start conferencing tools in correct sequence so as to avoid any potential software exception.

5.4.4.1 Synchronization between RAT and VIC

The audio and video involved in an on-going conference can be synchronized using VIC v.2.8 and RAT v.3.0.8 (or later). The enlarged image window offers a 'Mode' button offering "voice-switched" mode. Enabling this option will make the 'output' window to display audio transmission and video transmission details of active speakers only. To employ this, the conferencing tool should be started as follows [52],

Prompt> rat –lbl_channel x < IP address/port>

Prompt> vic –l x < IP address/port>

The command "lbl" or "l" means enabling 'lbl Conference Bus' support in both tools. And "x" is the number of Conference Bus channel in the range of 1 to 300, which is unique for all running sessions.

5.4.4.2 Synchronization between Agents

Agents should negotiate with each other in order to better synchronize conferencing tools. This issue becomes obvious in a sidebar. When a user initiates a sidebar, he/she

can start up the sidebar conference addressed by local IP. It is important that sidebar initiator's agent starts up RAT or VIC tool before the callee starts. Agents should communicate with each other in order to achieve this. Otherwise, code exception for starting RAT appears in callee's device. In order to make sure that the system is working properly, more communications are added between sidebar agents as shown below in figure 5.24. For example, a callee starts up RAT only after receiving 'confirmation' from a sidebar initiator.

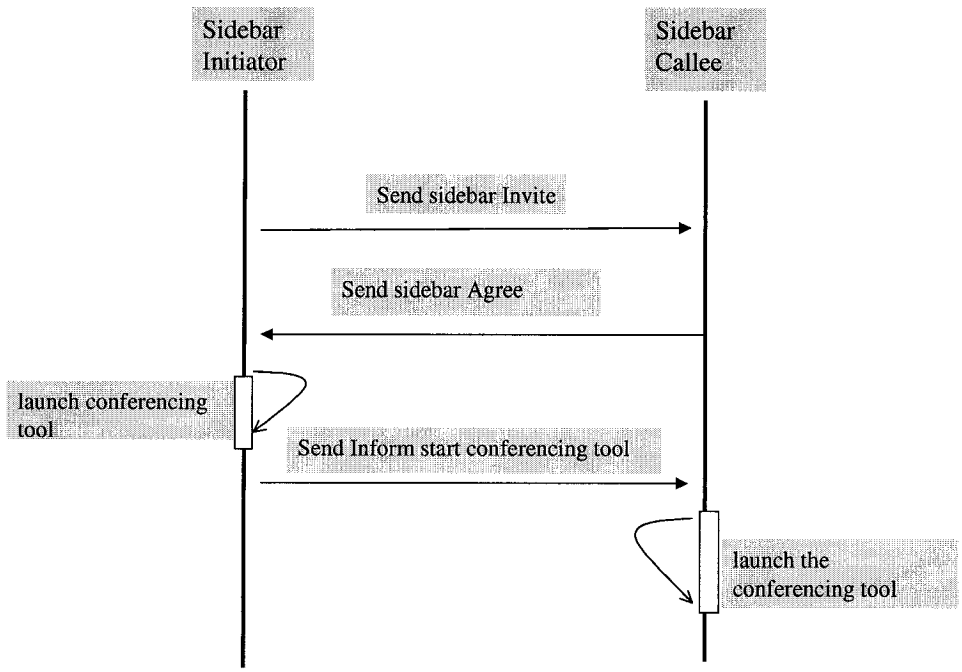


Figure 5.24 Agent's communication for the conference synchronization

5.5 Performance Evaluation

System evaluations are performed using scenarios similar to those described in sections 4.3 and 4.5. CA application written in java executes using J2SE [53] in a desktop PC. The

PC runs in Microsoft Windows 2000 operating system and its capabilities include 1.4 GHz AMD Athlon(tm) processor, 1 GB RAM, 10/100 Ethernet LAN support. Desktop PCs used in performance evaluation run distributed FIPA-OS agent platform. ACL message transmission between two desktop PC's needs approximately 1 second. Since the PCs used have high processing power, time spent by agents to process tasks is negligible compared to time spent on message transmission. VIC can be started in an instance and the time count is in milliseconds. However, RAT takes much longer time to start up. In this case it needs 2.6 seconds to start up.

CA application approximately takes 3 seconds to load and execute Java classes. After CA receives notification regarding the first user 'entrance', it starts processing immediately. It only takes 10ms for the CA to set up a conference state, prepare SIP INVITE media description, and try to send it out through SIP UA. The SIP UA takes approximately 3 seconds to send an INVITE to an 'entering' user, who will get a call prompt window in his/her screen. If this user accepts the call without much delay, his "200 OK" is quickly sent back to CA's UA. CA extracts information from "200 OK" SIP message and analyze it. Because SIP message sent from the socket is intercepted and then passed to CA for further processing, it takes CA longer time to handle this, approximately 8 seconds. Then CA updates the conference state and all related storages. While performing main ad-hoc conference tests, CA approximate takes 15 seconds to allow the user to formally join the conference and update all related data after receiving an 'user entering' notification.

When CA receives a SUBSCRIBE to a conference state, it only took 10 ms for CA to process the request and prepare the first notification to the user. The time spent by a user to receive the first notification about the conference state after he/she sends out his/her SUBSCRIBE is approximately 2 seconds. In the case of user leaving the room, CA takes only 15 ms to remove the user from the conference state and update the related storage. If all participants leave the room, the notification to clear the conference state only took 30ms for CA.

SAgent application takes approximately 2 seconds to load and execute Java classes. When receiving a sidebar request, SAgent takes 2 second to migrate sidebar agent to callee and another 2 seconds to migrate the agent to caller i.e., the sidebar initiator. Then, it took approximately 6 seconds for interaction between caller's and callee's sidebar agents to establish a sidebar session. So in average, it takes approximately 10 seconds for the SAgent to set up a new sidebar association.

5.6 Summary

This chapter presented the prototype implementation for the SIP-based ad-hoc conferencing management system. System prototype implementation details and different tools and techniques utilized in the implementation were elaborated.

Next chapter concludes the thesis by summarizing the work done in the project and gives suggestions for future research.

Chapter 6 Conclusions and Future Work

6.1 Conclusions

Communication over the Internet allows people to access global services afar. Internet Conferencing is a vital research topic which focuses on bringing physically separated people together by means of virtual conferencing instead of traditional face-to-face conferencing. This could save enormous invaluable time and energy and improve efficiency. To accommodate this growing amount of interaction requirements, SIP is designed by IETF to provide services and applications for multimedia communications over the Internet.

Agent technology is an emerging technology, which tries to reduce the workload of users and provide services to people by using autonomous software that could work on behalf of users. By combining SIP and agent technology, an ad-hoc conferencing management system has been designed and developed, keeping several constraints such as time, space, and location in mind. The strength of the system roots from proposed conferencing system architecture as well as developed conferencing processing engine, integrated system software, and service agents and their reusability.

The conferencing system architecture presented an SIP-based ad-hoc conferencing system using agents to facilitate service accessing and manage

communications among conference participants. Various services are provided in feature sets of the system. The room entity is an autonomous unit that coordinates actions among users and between users and services. System behavior is determined by policies. Agents work on behalf of the room entity and interact with other agents to manage the ad-hoc conference and handle service requirements from users. The whole implementation was based on powerful software engineering techniques, and it is open for future extensibility.

Leading-edge technologies have been utilized to implement the system. Specifically, JAVA environment, FIPA-OS agent platform, SIP and conferencing tools RAT and VIC have been utilized to accomplish implementation tasks. The integration of system modules was achieved smoothly.

6.2 Future Work and Suggestions

A lot of work can be contributed in future to further refine the system. Architecture of the SIP-based ad-hoc conferencing system for one Internet site is the basis for multi-site conferencing system over the Internet and it is the most important part of designing a sophisticated multimedia conferencing system over the Internet. The thesis focused on system model development at one Internet site. However, this SIP-based ad-hoc conferencing model can be further modified to encompass multiple site conferencing over the Internet. The capability of the conferencing system can also be enriched with more features and thereby the conferencing system could become more effective and adaptive to the Internet. To adapt the system from one Internet site to multi-site, new

scenarios need to be designed to satisfy characteristics such as autonomy and flexibility of the whole system. Further modifications and adaptations to one site system are required, due to conflictions and requirements of multi-room system. Research related to a multi-room system needs further investigation.

In addition, it is also possible to add scenarios, wherein, even though users are physically outside of any conferencing room, they can join an on-going conference through the Internet. For example, a user currently outside of any conferencing room may join the ad-hoc conference through the Internet and could virtually become a member of one room, and thus could join multi-room conferencing and access resources in the conferencing rooms. How to add these scenarios to the conferencing system architecture could be another interesting research topic.

Issues related to CPL policy scripts such as SIP REGISTER payload can be addressed in detail as well. The scenario using CPL and SIP REGISTER for *source policy* definition has been designed and can further be implemented. Furthermore, research can investigate user requirements and produce better tailored personal policies by putting the most important information into CPL scripts, so as to improve personalized environment for users in a conferencing room. Currently CPL has defined features for describing Internet telephony services such as call forward and call screening [12]. In order to describe users' different constraints in *source policy* definition, new CPL features can be exploited beyond those listed for fully facilitating users' requirements. To achieve this, CPL extensions should be developed for administration and control. Finally,

with the new release of SIP extensions, it is also possible to adopt a new mechanism to design and develop *source policy* definition.

In general, a SIP-based ad-hoc conferencing management system has been designed which is open to new features and modifications. Any services other than those mentioned in the thesis can be added to the system to enrich capability of the conferencing system and enhance system's functionalities.

Appendix A

A.1 SIP REFER – Call Transfer [18]

SIP has a simple mechanism to realize call transfer by means of REFER method. In the REFER method, the recipient of REFER request is indicated to contact a third party using the contact information provided in this method.

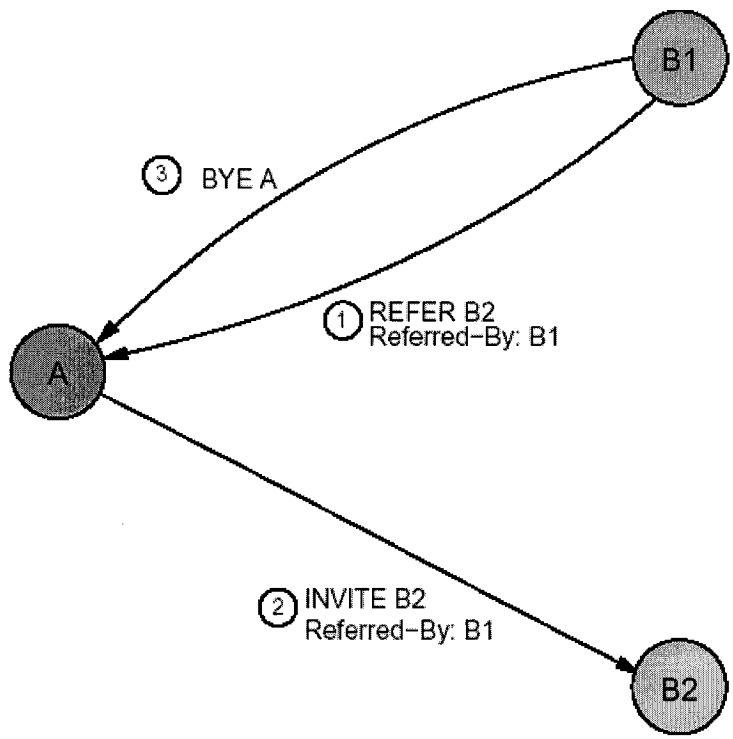


Figure A.1 Call transfer [14]

As showed above in figure A.1, user 'B1' sends a REFER request to user 'A' (1), indicating that the user 'A' should contact user 'B2'. This causes user 'A' sending 'B2' a regular SIP INVITE request to negotiate a new session between user 'A' and user 'B2'

(2). After a session is established between user 'A' and user 'B2', a BYE is sent from user 'B1' to user 'A' to terminate the session between them (3).

A.2 SIP 3PCC – Third Party Call Control [20]

Third-party call control in SIP allows one entity (called controller) to set up and manage communication relationships between two or more other parties.

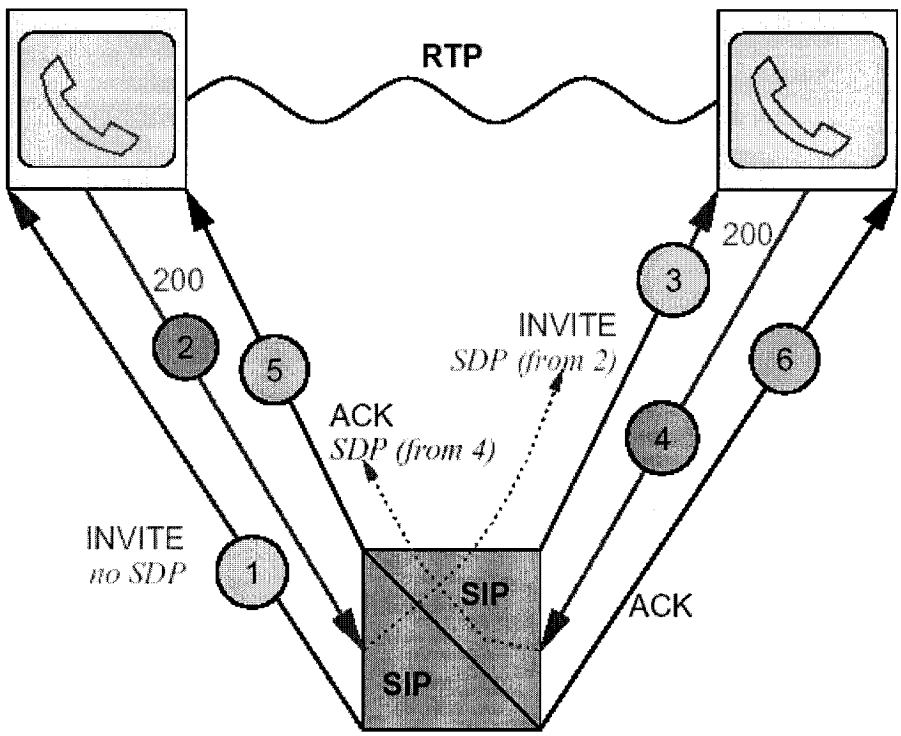


Figure A.2 Third-party call control [14]

As shown above in figure A.2, the controller first sends an INVITE to user 'A' (1). After the controller gets "200 OK" response from user 'A' (2), it sends another

INVITE to user 'B' with same SDP session description as in "200 OK' received from user 'A' (3). After the controller gets response from user 'B' (4), it sends acknowledgement signal to both parties (5) (6). When the acknowledgement is complete, the controller has set up a media session between these two users. By means of RTP [3], user 'A' and 'B' can have media streams transmitted between them.

References

[1] IETF home page, available at http://www.ietf.org.

[2] M. Handley, H. Schulzrinne, E. Schooler and J. Rosenberg, "SIP: Session Initiation Protocol," Request for Comments 2543, Internet Engineering Task Force, Mar. 1999.

[3] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: a transport protocol for real-time applications," Request for Comments (Proposed Standard) 1889, Internet Engineering Task Force, Jan. 1996.

[4] M. Handley and V. Jacobson, "SDP: session description protocol," Request for Comments (Proposed Standard) 2327, Internet Engineering Task Force, Apr.1998.

[5] E. M. Schooler, S. L. Casner, and J. Postel, "Multimedia conferencing: Has it come of age?," in *Proceedings of the 24th Hawaii International Conference on System Science*, vol. 3, (Hawaii), IEEE, Jan. 1991, pp. 707-716.

[6] T.Magedanz and A. Karmouch (Guest Editors), "Mobile Software Agents for Telecommunication Applications", *Journal of Computer Communication*, Vol. 23, Issue. 8, 2000.

[7] FIPA Agent Software Integration Specification (XC00079B), Aug. 2001, http://www.fipa.org/specs/fipa00079/XC00079B.html

[8] B. Chaib-draa and F. Dignum, "Trends in Agent Communication Language", *Computational Intelligence*, 18(2), 2002, pp. 89-101.

[9] J. Rosenberg, "A Framework for Conferencing with the Session Initiation Protocol", Internet draft, Internet Engineering Task Force, May 1, 2003. Work in progress.

[10] S. Bessler, A. V. Nisanyan, K. Peterbauer, R. Pailer, J. Stadler, "A Service Platform for Internet-Telecom Services using SIP", *Smartnet 2000*

[11] Jonathan Rosenberg, Jonathan Lennox and Henning Schulzrinne, "Programming Internet Telephony Services", *IEEE Internet Computing Magazine*, Vol. 3, No. 3, May/June 1999, pp. 63-72, and *IEEE Network Magazine*, vol. 13, No. 3, May/June 1999, pp. 42-49.

[12] J. Lennox and H. Schulzrinne, "CPL: A Language for User Control of Internet Telephony Services," Internet Draft, Internet Engineering Task Force, Jan. 2002. Work in progress.

[13] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," Request for Comments: 2068, Jan.1997.

[14] Henning Schulzrinne, "The Session Initiation Protocol (SIP)", available at http://www.cs.columbia.edu/~hgs/teaching/ais/slides/sip_long.pdf. Slides.

[15] Henning Schulzrinne, Jonathan Rosenberg, "The IETF Internet Telephony Architecture and Protocols", available at http://www.computer.org/internet/telephony/w3schrosen.htm

[16] J. Rosenberg *et al.*, "SIP extensions for presence," Internet Draft, Internet Engineering Task Force, June 2000. Work in progress.

[17] Jonathan Rosenberg, Dean Willis, Robert Sparks, Ben Campbell, Henning Schulzrinne, Jonathan Lennox, Bernard Aboba, Christian Huitema, David Gurle, Dave Oran, "SIP Extensions for Instant Messaging", Internet Draft, Internet Engineering Task Force, Feb. 2001. Work in progress.

[18] R. Sparks, "SIP Call Control – Transfer", Internet Draft, Internet Engineering Task Force, Feb. 2001. Work in progress.

[19] H. Schulzrinne and J. Rosenberg, "SIP Call Control Services," Internet draft, Internet Engineering Task Force, Feb. 1998. Work in progress.

[20] Jonathan Rosenberg, Jon Peterson, Henning Schulzrinne, Gonzalo Camarillo, "Third Party Call Control in SIP", Internet Draft, Internet Engineering Task Force, Mar. 2001. Work in progress.

[21] E. Shapiro, "Virtual places – a foundation for human interaction," in *Proc. Of the Second World Wide Web Conference'94*, Chicago, Illinois, Oct, 1994.

[22] Crowley, T., Milazzo, P., Baker, E., Forsdick, H., Tomlinson, R., "MMConf: An Infrastructure for Building Shared Multimedia Applications", *Proceedings CSCW '90*, Los Angeles, CA, Oct. 1990, pp. 329-342.

[23] Forsdick, H.C., "Explorations into Real-time Multimedia Conferencing", *Proceedings 2nd International Symposium on Computer Message Systems*, Sept. 1985, pp. 299-315.

[24] Root, R.W., "Design of a Multi-Media Vehicle for Social Browsing", *Proceedings CSCW '88*, Portland, OR, Sept. 1988, pp. 25-38.

[25] Casner, S., Seo, K., Edmond, W., Topolcic, C., "N-Way Conferencing with Packet Video", *Proceedings 3rd International Workshop on Packet Video*, VISICOM '90, Morristown, NJ, Mar. 1990.

[26] Schooler, E.M., Casner, S.L., "A Packet-switched Multimedia Conferencing System", *ACM SIGOIS Bulletin*, Jan. 1989, pp.12-22.

[27] P. Lantz, "Usage of H.323 on the Internet," Internet Draft, Internet Engineering Task Force, Feb. 1997. Work in progress.

[28] Henning Schulzrinne, Jonathan Rosenberg, "The IETF Internet Telephony Architecture and Protocols", available at http://www.computer.org/internet/telephony/w3schrosen.htm

[29] S. Corson, J. Macker, "Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations", Request for Comments 2501, Jan. 1999.

[30] A. Bruce McDonald and Taieb Znati, "A Mobility-Based Framework for Adaptive Clustering in Wireless Ad-Hoc Networks", *IEEE Journal on Selected Areas in Communication*, Vol. 17, No. 8, Aug. 1999, pp. 1466-1487.

[31] F. Vakil, A. Dutta, J-C. Chen, M. Tauil, S. Baba, N. Nakajima, Y. Shobatake, H. Schulzrinne, "Supporting Mobility for Multimedia with SIP," Internet draft, Internet Engineering Task Force, Nov. 2000. Work in progress.

[32] Stan Moyer, Dave Marples, and Simon Tsang, "A Protocol for Wide-Area Secure Networked Appliance Communication," *IEEE Communications Magazine*, Oct. 2001, pp. 52-59.

[33] H. Schulzrinne, "A comprehensive multimedia control architecture for the Internet", *Proc. Of the Int. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, St. Louis, Missouri, May 1997.

[34] K. Singh, G. Nair, and H. Schulzrinne, "Centralized conferencing using SIP," in *Internet Telephony Workshop 2001*, New York, Apr. 2001

[35] Wenyu Jiang, Jonathan Lennox, Henning Schulzrinne and Kundan Singh, "Towards Junking the PBX: Deploying IP Telephony," In *ACM International*

Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), Port Jefferson, New York, June 2001.

[36] Henning Schulzrinne and Elin Wedlund , "Application-Layer Mobility using SIP , " *ACM Mobile Computing and Communications Review,* Volume 4, Number 3, July 2000, pp. 47-57.

[37] J.Rosenberg, H.Schulzrinne, "Models for Multi Party Conferencing," Internet Draft, Internet Engineering Task Force, Nov. 2000. Work in progress.

[38] Henning Schulzrinne and Jonathan Rosenberg, *Signaling for Internet Telephony* , Columbia University Technical Report CUCS-005-98, New York, N.Y., 1998.

[39] D. Trossen (Editor), "Conference Course Control Problem Statement", Internet Draft, Internet Engineering Task Force, Nov. 2001. Work in progress.

[40] International Telecommunication Union, "Visual telephone systems and equipment for local area networks which provide a non-guaranteed quality of service," Recommendation H.323, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, May 1996.

[41] J. Van Dyke, E. Burger, "SIP URI Conventions for Media Servers," Internet draft, Internet Engineering Task Force, January 2002. Work in progress.

[42] J. Lennox and H. Schulzrinne, "Transporting user control information in SIP REGISTER payloads," Internet Draft, Internet Task Force, Mar. 1999. Work in progress.

[43] J. Rosenberg,H. Schulzrinne, "SIP Event Packages for Call Leg and Conference State," Internet Draft, Internet Engineering Task Force, 1 Mar., 2002. Work in progress.

[44] FIPA ACL Message Structure Specification (XC00061D), August 2000, available at http://fipa.comtec.co.jp/fipa/spec/fipa00061/XC00061D.pdf

[45] Robust Audio Tool (RAT), University College London, available at http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/

[46] Videoconferencing Tool (VIC), Network Research Group at the Lawrence Berkeley National Laboratory in collaboration with the University of California, Berkeley, available at http://www-mice.cs.ucl.ac.uk/multimedia/software/vic/

[47] FIPA-OS SourceForge Home Page, available at http://fipa-os.sourceforge.net/

[48] Poslad S. J., Buckle S.J., and Hadingham R., "The FIPA-OS agent platform: Open Source for Open Standards", *Proceedings of PAAM 2000*, Manchester UK, Apr. 2000, pp. 355-368.

[49] *FIPA-OS user guide,* version 2.1.0, Emorphia Research, Emorphia Ltd., available at http://sourceforge.net/project/showfiles.php?group_id=3819

[50] SIP User Agent and SIP server (sipc & sipd), Columbia University, http://www1.cs.columbia.edu/~xiaotaow/sipc/

[51] *Robust Audio Tool (RAT) user guide*, version RAT 4.2, University College London, available at http://www.vrvs.org/Doc/Applications/rat4-userguide.pdf

[52] *Videoconferencing Tool (VIC) user guide*, version 2.8, University College London, available at http://www.vrvs.org/Documentation/Applications/vic-userguide.pdf

[53] Java 2 Platform, Standard Edition (J2SE), available at http://java.sun.com/j2se.