Internet Telephony Services for Presence With SIP and Extended CPL

Dongmei Jiang

Thesis submitted to the

Faculty of Graduate and Postdoctoral Studies

in partial fulfillment of

the requirements for the degree of

Master of Computer Science

Under the auspices of the

Ottawa-Carleton Institute for Computer Science

University of Ottawa

Ottawa, Ontario, Canada

December 2003

Abstract

Internet Telephony is the next generation of telephony with many new features and low cost. Because of the explosion of new features, it has become critical to control and manage these features. The main challenge in Internet Telephony is service programming. The Call Processing Language (CPL) is a solution for end users to describe and control their services in Internet Telephony. Current CPL focuses on call processing services only. It is not adequate for the definition of many types of new services, such as the combination of telephony services with email, instant messaging, presence etc. This thesis extends CPL to describe new Internet Telephony services including presence services and call processing services related to presence. In the thesis, the presence system is systematically described in a three-layer architecture. End user's presence services and system basic services are clearly separated in the architecture. Presence information, as the basis of presence services, is extended from traditional "online" and "offline" indicators to include broader meaning, such as location, phone line status, role and availability status etc. Through CPL extensions for presence, user's new presence services and new presence related call processing services are illustrated by using various examples. A simulation system is implemented to demonstrate the Internet Telephony services specified in extended CPL. End users can create and modify their own services via the Graphic User Interfaces (GUIs) and access their services at any location through the Internet. The simulation system is verified with various test cases.

Keywords: Feature, Service, Policy, Internet Telephony, Presence, SIP, CPL, PIDF

Acknowledgements

I would like to thank my supervisor, Dr. Luigi Logrippo, for his guidance with this research. His encouragement and support are very much appreciated.

I would like to thank all members in the LOTOS group at the University of Ottawa, in particular Jacques Sincennes, Romelia Plesa, and Yiqun Xu, for their constructive discussions.

I want to thank Dr. Ramiro Liscano and Mr. Tom Gray for their precious advices and suggestions. The initial idea for this work was given to me by Dr. Liscano.

I appreciate the financial support from CITO, Mitel, NSERC, Ontario Ministry of Training, Colleges and Universities, and the University of Ottawa.

Finally, I would like to thank my husband, Ping, my son, Dajian, for their understanding and patience throughout the years.

Table of Contents

Acronyms

API	Application Programmer's Interface
BS	Base Station
BSC	Base Station Controller
CPIM	Common Profile for Instant Messaging
CRLF	Carriage-Return Line-feed
CPL	Call Processing Language
CPP	Common Profile for Presence
DNS	Domain Name Server
DTD	Document Type Declaration
GUI	Graphic User Interface
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IETF	Internet Engineering Task Force
IN	Intelligent Network
ITU	International Telecommunications Union
JDBC	Java Database Connection
JPEG	Joint Photographic Experts Group
MIME	Multipurpose Internet Mail Extension
MS	Mobile Station
MSC	Mobile Station Switch Center
PA	Presence Agent
PDA	Personal Digital Assistant
PIDF	Presence Information Data Format
PLMN	Public Land Mobile Network
P-PA	Presentity Side Presence Agent
P-SIP	Presentity Side SIP
PSTN	Public Switched Telephone Network
PUA	Presence User Agent

RTCP	Real Time Control Protocol
RTP	Real Time Protocol
SCP	Service Control Point
SIP	Session Initiation Protocol
SMTP	Simple Mail Transfer Protocol
SSP	Service Switching Point
TCP	Transmission Control Protocol
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W-PA	Watcher Side Presence Agent
W-SIP	Watcher Side SIP
XML	Extensible Markup Language

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Background and Motivation

1.1.1 Internet Telephony

According to the International Telecommunications Union - Telecommunications Standardization Sector's (ITU-T's) definition [1], a **service** is offered by an administration to its customers in order to satisfy a specific (set of) telecommunication requirement(s), while a **feature** is the smallest part of a service that can be perceived by the service user. A **policy** is a specific service or a specific feature that can be defined by a user. In actual usage and in most references, feature and service are not strictly distinguished. Generally speaking, service is a broader term, which may include a set of features. Feature is defined to be the smallest functional unit that can be sold to users.

Telephone business started from basic call connection services with in-band signaling that are not efficient and not secure. Then out-of-band signaling, i.e. common channel signaling that separates signaling channels from voice channels, totally changed the architecture of the telephony system. SS7 [2], as an example of common channel signaling protocol, makes the system more efficient so that many advanced telephony features can be offered. After that, the intelligent network (IN) [3] was developed. IN utilizes the elements in SS7 and offloads much intelligence from switches i.e. from Service Switch Points (SSPs) to Service Control Points (SCPs). Due to the use of powerful SCPs, new services were created more easily and call processing was handled more efficiently. Later on, much effort was put in migrating voice from circuit-switched networks to packet-switched networks. In recent years, **Internet Telephony** [4] has become a much more attractive solution. It enhances existing telephony features and creates a number of new features with the integration of Internet services. Internet

Telephony is considered as the next-generation uniform communication model, which will finally phase out the Public Switched Telephone Network (PSTN) [5].

One major difference between PSTN and the Internet Telephony is that the Internet Telephony signaling protocol, Session Initiation Protocol (SIP) [6], is built on top of existing data communication network. This makes it natural and easy to combine voice services and data services. This combination results in many new features that are not realizable in PSTN. Internet Telephony can have PSTN features together with computer features and Internet features. Some of the advantages of using Internet Telephony are as follows:

Low-cost voice calls: Internet users usually pay flat fees for monthly access. On the Internet, there is actually no such concept of long-distance as in PSTN. High long-distance call fees will be dramatically reduced. The main reason is that in the Internet there is no resource reservation as in PSTN.

Sound grading: PSTN has only one sound quality (4KHz). Internet can have many levels of sound quality as long as bandwidth is available.

Video telephony: Real time video delivery has been already achieved on the Internet. This could be easily adapted to realize video telephony.

More powerful call processing: An end user can program his specific services to process his calls. These services can be based on time, language, priority, address etc. They can also be based on a user's presence status (i.e. the user's ability or willingness to communicate). A user can reject anonymous calls and can forward his incoming calls to his voice mail if he is not available at the moment.

Web-based call centers: SIP addresses can be embedded in web pages. Customers could initiate calls just by clicking on those SIP numbers. While they are talking with the service staff, they can browse the web pages at the same time. Also the service staff could

feed customers relevant information pages or guide the customers to make orders step by step.

Real-time billing: With PSTN, a user can only check his bill when the bill comes to his mailbox next month. Real-time billing is just not possible since billing process is so complicated in PSTN. With Internet Telephony, the end devices (computers) have enough computational power, which enables them to access the billing gateway directly to get the billing information in a real-time fashion.

Much industrial and research interest has been generated by the developments in Internet Telephony. These developments are a complete change compared to PSTN. The key point of Internet Telephony is the provision of new services. It can combine telephony services with web, email, instant messaging, presence, text chat, interactive games etc. Because of the explosion of new features, it has become critical to control and manage them. The main challenge in Internet Telephony is service programming [7]. The **Call Processing Language** (CPL) is a solution for end users to describe and control their services in Internet Telephony [8].

1.1.2 Programming Internet Telephony Services in CPL

Due to its safe, efficient, flexible, simple and extensible properties, CPL is designed for end users to describe and control services in Internet Telephony [9]. As a programming language, CPL only contains switches and triggers to perform different actions. However, CPL is not a Turing-complete language, it dose not provide loops or recursion and it cannot call external programs. It does not have its own variables and can only access limited resources for safety considerations. End users can create and update CPL scripts and enable them at any time. CPL uses the syntax of the Extensible Markup Language (XML) [10], which makes CPL easy to extend because of the simplicity and extensibility of XML.

CPL is signaling protocol independent, which means that it can work either with H.323 [11] or with SIP. In this thesis, only the SIP signaling protocol is used. CPL is based on SIP and CPL scripts reside on SIP servers or intelligent agents to describe and control end user services.

SIP is defined in RFC2543 (March 1999) [12] and modified in RFC3261 (June 2002) [13] as "an application-layer control protocol that can establish, modify and terminate multimedia sessions or calls". The basic information about SIP will be reviewed in chapter 2, which is necessary to understand CPL. Based on SIP, CPL can be discussed in detail, such as where the services live, what the programs can control, when the programs are executed, what information the programs provide, what resources the programs can have access to, who can create the programs, how the programs are instantiated, etc.

1.1.3 CPL Limitations

The major advantage of Internet Telephony is the provision of new services. Current CPL is not adequate for the definition of many types of desirable new services since it focuses on call handling services only. It cannot describe services that combine web, email, instant messaging, presence, text chat etc. These limitations can be remedied by extending CPL. CPL and CPL extensions together can describe many Internet Telephony features.

1.2 Thesis Objectives

Presence, also known as presence information, conveys the willingness and ability of a user to communicate across a set of devices with others on the network [14]. RFC 2778 [15] defines a model and terminology for describing systems that provide presence information. In the model, a presence system is a presence service that accepts, stores, and delivers presence information to the interested parties.

Many new services in Internet Telephony are the combination of telephony services with presence services. In order to describe these combined services, this thesis needs to start by describing presence systems and presence services. After that, it should propose an approach to allow end users to program these services by themselves. A demonstration system is the best way to show these new services in Internet Telephony. The system should allow end users to write their specific services easily through Graphic User Interfaces (GUIs). The specific objectives of the thesis are listed as follows:

Objective 1: Describe presence system

The current research on presence is just at the beginning. There are no complete definitions or descriptions of presence systems. In order to describe the combination of telephony services with presence services, the first objective is to systematically describe presence systems and explain the basic concepts of such systems. We need to clarify the relation of SIP services and presence services. In order to describe end user specific presence services, we need to separate presence system services from end user services.

Objectives 2: Extend presence information

Presence information is the basis for presence services. In the past, presence was only limited to "on-line" and "off-line" indicators. The notion of presence can be much broader. It can include the location of a user, the role that the user is currently taking or the user's willingness to communicate etc. These extensions of presence information will much enrich presence related services. We need to explore the basic procedures to extend presence information. We need to explain how to define the extensions of presence information, how to write presence documents, how to declare the extensions in the presence documents, etc.

Objective 3: Describe user specific presence related services

We need a language to describe end user specific presence related services. Such a language can be obtained by extending CPL for presence. In this way, presence services as well as call processing services related to presence can be specified in extended CPL naturally.

Objective 4: Demonstrate Internet Telephony services

We need to implement a simulation system to demonstrate Internet Telephony services, which include presence services, call handling services and their combinations. These end user specific services are specified in extended CPL. End users should be able to create and modify these services through Graphic User Interfaces (GUIs).

1.3 Thesis Organization

Chapter 1: Introduction

This chapter reviews the development of telephony features and illustrates new features in Internet Telephony. CPL and its purpose are introduced. The objectives and organization of the thesis are described.

Chapter 2: Internet Telephony Services with SIP and CPL

This chapter reviews Internet Telephony services with SIP and CPL. Basic concepts of SIP and CPL are described. Personal mobility, a major feature of SIP, is discussed. As well, **service mobility**, which is the characteristic of Internet Telephony that allows users to find the same services anywhere, has been introduced.

Chapter 3: Presence System

Presence systems and the basic concepts of presence systems are introduced in the framework of a three-layer architecture. The location of end user specific services is identified in the architecture. Basic services of presence systems are described through various scenarios. How SIP backs up presence services is illustrated. The limitations of basic services of presence systems are discussed.

Chapter 4: Extensions of Presence Information and CPL

This chapter describes presence related services for end users. These services are specified in extended CPL. The reasons why extended CPL is chosen to describe end user

services in presence systems are discussed. How to extend presence information and how to extend CPL for presence are described in detail. With CPL extensions, presence services, call handling services related to presence and their combinations are illustrated through various examples.

Chapter 5: Simulation System

The Java implementation of the system described in chapter 3 and chapter 4 is discussed in this chapter. The simulation system is capable of demonstrating presence services, call processing services and their combinations. These services, specified in extended CPL can be created and modified by end user themselves through Graphic User Interfaces (GUIs).

Chapter 6: Summary and Future Work

This chapter gives the conclusion of the thesis and discusses the potential future work.

1.4 Related Work

Internet Telephony Signaling Protocol, SIP:

Currently, there are two main sets of standards for Internet Telephony [11]. One standard is the Session Initiation Protocol (SIP) provided by the Internet Engineering Task Force (IETF), the group that standardizes protocols used on the Internet. The other standard is the H.323 suite of protocols developed by the International Telecommunication Union-Telecommunications Standards Sector (ITU-T). Both standards provide mechanisms for customer call establishment, modification and teardown to support advanced services. According to [11], compared to H. 323, the SIP protocol is more flexible in adding new advanced services.

The SIP standard was first published as IETF RFC2543 [12] in 1999 and was updated by RFC3261 [13] in June 2002. Later, SIP extensions for presence were added to the SIP standard by IETF Draft [14] in January 2003. A book on the SIP standard [28] was published in 2001.

End User Service Description and Controlling, CPL:

The key point in Internet Telephony is service programming. The Call Processing Language (CPL) framework and requirements were established by IETF RFC 2824 [18] in May 2000. The definition of CPL (version 6) was published in IETF Internet Draft [9] in January 2002 and was updated by Internet Draft [27] in August 2003. The current research work on CPL will be reviewed in chapter 2 in detail.

CPL Extensions for Presence:

The research on presence system is a very new topic and no journal papers exist. A model for presence and instant messaging was established by IETF RFC 2778 [15] in February 2000. A Presence Event Package for SIP was defined in IETF Internet Draft [14] in January 2003. Presence Information Data Format (PIDF) was published in IETF Internet Draft [21] in May 2003.

XiaoTao Wu was the first researcher who proposed the idea to extend CPL for presence and published the idea in IETF Internet Draft [16]. He added the capabilities to describe presence system services and the focus was on the basic system services instead of end user specific services.

Based on his work, we have systematically described basic concepts of presence systems and their services in a three-layered architecture. System services and end user specific services are clearly separated. In order to be able to provide more powerful presence services, we have extended presence information. As well, we have added more capabilities in CPL extensions for presence so that we can describe new presence services and call handling services, which are based on presence information.

Chapter 2 Internet Telephony Services with SIP and CPL

2.1 Introduction

This chapter introduces Internet Telephony services with the Session Initiation Protocol (SIP) and the Call Processing Language (CPL). Section 2.2 introduces basic concepts of SIP and discusses personal mobility, a major feature of SIP. Section 2.3 introduces basic concepts of CPL and explains how to use CPL to describe end user services. Section 2.4 introduces **service mobility**, which is the characteristic of Internet Telephony that allows users to find the same services anywhere. Section 2.5 discusses CPL limitations and section 2.6 is the conclusion of this chapter.

2.2 Internet Telephony Features with SIP

Internet Telephony refers to real-time voice or multimedia communications that are transported via the Internet. The signaling protocol SIP plays a most important role in Internet Telephony. SIP can establish, modify and terminate multimedia sessions and calls. With SIP, it is possible to create many new services with the integration of telephony services with web, email, instant messaging, presence, text chat, interactive games, etc. The creation of new features depends on the power of the signaling protocol, SIP.

2.2.1 SIP is a Signaling Protocol

SIP is defined in RFC2543 (March 1999) [12] and modified in RFC3261 (June 2002) [13]. SIP is defined as "an application-layer control protocol that can establish, modify and terminate multimedia sessions or calls".

SIP is modeled on two important protocols, Simple Mail Transfer Protocol (SMTP) and Hypertext Transfer Protocol (HTTP). The use of textual messages makes SIP simple and efficient.

2.2.2 SIP is on Application Level

SIP is defined on top of transport layer, as shown in Figure 2.1. It can use either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) to send messages. This provides much flexibility of implementing different services according to the different requirements. SIP messages can convey arbitrary signaling payload: session description, instant messages, presence document, Joint Photographic Experts Groups (JPEGs) and any MIME (Multipurpose Internet Mail Extensions) type. Complex services could be built on top of SIP, for example, a CPL script to reject calls from unknown callers (see section 2.4).

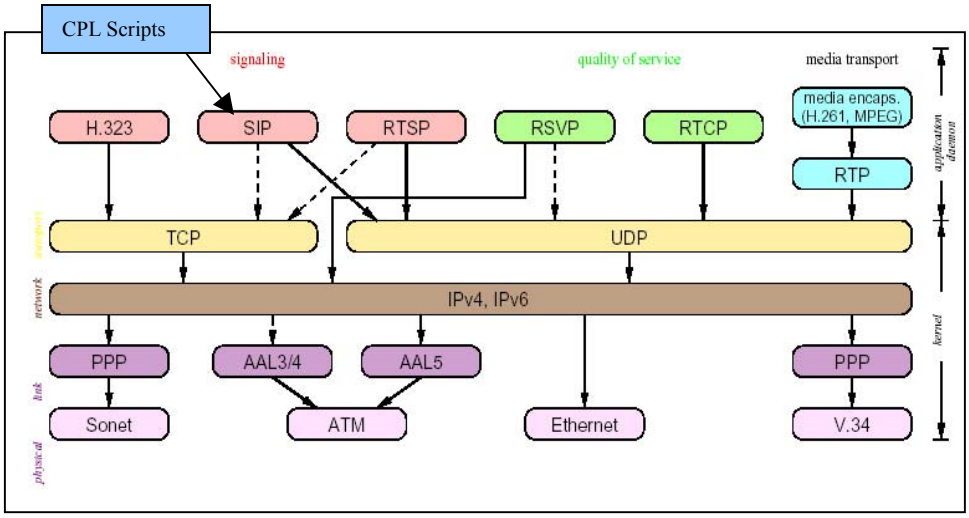

Figure 2.1 Internet Telephony Protocols [17]

SIP is just one of the protocols supporting Internet Telephony. From the functional point of view, SIP mainly readies the IP addresses and port numbers ready to establish a session. The media/data packets, which are delivered and controlled by other protocols such as Real Time Protocol (RTP) and Real Time Control Protocol (RTCP), most probably will take different paths.

2.2.3 SIP Components

From the architectural point of view, SIP works as a client/server model, just like HTTP. A SIP system has two types of components: user agents and network servers. From a protocol point of view, there are two types of user agents, **user agent client** (UAC) and **user agent server** (UAS). UAC works for callers to initiate the calls and UAS works for callees to answer the calls. Between UAC and UAS, there may exist two different servers: proxy servers and redirect servers. A **proxy server** acts on behalf of the UAC and forwards the request to UAS or other servers. A **redirect server** receives the requests and responds to the client directly to tell it which server it should contact next. The whole SIP system is configured so that all these operational details are transparent to end users.

The **registrar** is another important SIP server. It accepts REGISTER request from clients. This registration information is the basis to provide location service, which is the key to achieve mobility as will be discussed in section 2.2.5.

2.2.4 SIP Methods and Messages

SIP has two types of messages: request and response. Request messages are specified in six basic SIP methods defined in RFC2543 [12] (see Table 2.1). Response messages are specified in six classes of response codes (see Table 2.2).

Method	Description
INVITE	Request to set up a session
ACK	Message from client to indicate that a successful response to an INVITE has been received
OPTIONS	A Query to a server about its capabilities
BYE	A call is being released by either party
CANCEL	Cancel any pending requests. Usually sent to a Proxy Server to cancel searches
REGISTER	Used by client to register a particular address to the SIP server

Table 2.1 SIP Request Methods

Code	Description	Examples
1xx	Informational – Request received, continuing to process request	180 Ringing 181 Call is Being Forwarded
2xx	Success – Action was successfully received, understood and accepted	200 OK
3xx	Redirection – Further action needs to be taken in order to complete the request	300 Multiple Choices 302 Moved Temporarily
4xx	Client Error – Request contains bad syntax or cannot be fulfilled at this server	401 Unauthorized 408 Request Timeout
5xx	Server Error – Server failed to fulfill an apparently valid request	503 Service Unavailable 505 Version Not Supported
6xx	Global Failure – Request is invalid at any server	600 Busy Everywhere 603 Decline

Table 2.2 SIP Response Status Codes [13]

An example of a SIP request message is displayed in Table2.3. The message consists of a start line, one or more header fields (headers), empty line, carriage-return line-feed (CRLF) (indicating the end of the headers), and an optional message body.

	Message	Description
Start Line	**INVITE sip: uB@lucent.com** SIP/2.0	METHOD, URL
Headers Message	Via: SIP/2.0/**UDP lucent.com: 4545** From: User A <sip: **dongmei@site.uottawa.ca**> To: User B <sip:**sharon@site.uottawa.ca**> Call-ID: **34567@uottawa.ca** Cseq: **1 INVITE** Content-Type: **application/sdp** Content-Length: **187**	Protocol, host:port from_user@source to_user@destination localid@host seq# method type of body media length of body
Body	v=0 o=**user1 53655765 2353687637** IN IP4 **128.3.4.5** c=IN IP4 **224.2.0.1/127** m=**audio 3456** RTP/AVP **0**	v=0 origin_user, timestamp, timestamp …… host destination address media type, port, …… payload types

Table 2.3 A SIP Request Message

The SIP response message is similar to the SIP request message except the start line. The start line is a request-line for a request message and it is a status-line for a response message. In the status-line, the SIP version is followed by a response code. For an example, status-line, “SIP/2.0 200 OK”, indicates the SIP response message is “200 OK”

written in SIP version 2.0. “200 OK” means the corresponding SIP request message is successfully received, understood and accepted.

2.2.5 SIP Features

There is no doubt that all the telephony features realized already in PSTN can be realized with SIP [17]. Moreover, SIP could introduce a whole line of new services on the basis of functionalities such as call forking, personal mobility, integration of computers and networks etc. Most these new services are definitely not possible in PSTN. Many added-value services have more web flavor than telephony flavor. Mobility deserves more efforts to discussion since it is involved in many advanced features. We can classify mobility into three different categories:

Terminal Mobility: Terminals can move among sub networks (wireless cell phones).

Personal Mobility: A person can use different devices and different addresses. The network identifies the user by its unique identification rather than by terminals (SIP phones).

Service Mobility: A user can have the same services from different locations and devices (Internet telephones: services mobility through CPL will be discussed in section 2.5).

Both Public Land Mobile Network (PLMN) wireless telephony and Internet telephony with SIP can realize mobility, but in different ways. In the PLMN, the mobility is Terminal Mobility. The network recognizes wireless phones/mobile stations. The system consists of Mobile Stations (MS), Base Stations (BS), Base Station Controllers (BSC) and Mobile Station Switch Centers (MSC) to support mobility feature. Handoff procedure must be handled carefully. This is really a complicated system. With SIP instead, the mobility is achieved by the user’s registration to SIP servers using the REGISTER method. The network locates and recognizes the user through its location service based on the information in the Registrar server. SIP users could log on several

different end devices with different addresses at different locations at the same time. As long as he/she registers with the same SIP identification, the network knows where the user can be reached. This is personal mobility. Service mobility will be discussed in section 2.5.

Personal Mobility allows Internet telephony users to be mobile with one published SIP address. Customers can use different phones or computers at the same or different locations. Fig 2.2 shows an example of personal mobility from [17] and Fig. 2.3 shows the possible scenario in the form of a message sequence chart. The numbers in Fig. 2.3 are corresponding to the step numbers in the Fig. 2.2.

Suppose that user Bob works at Lucent and has an office at Lucent Technologies. He has published a single SIP address, *bob@lucent.com*, which is registered in the Lucent SIP server. Bob is also an assistant professor at Columbia University, where he has a lab and an office.

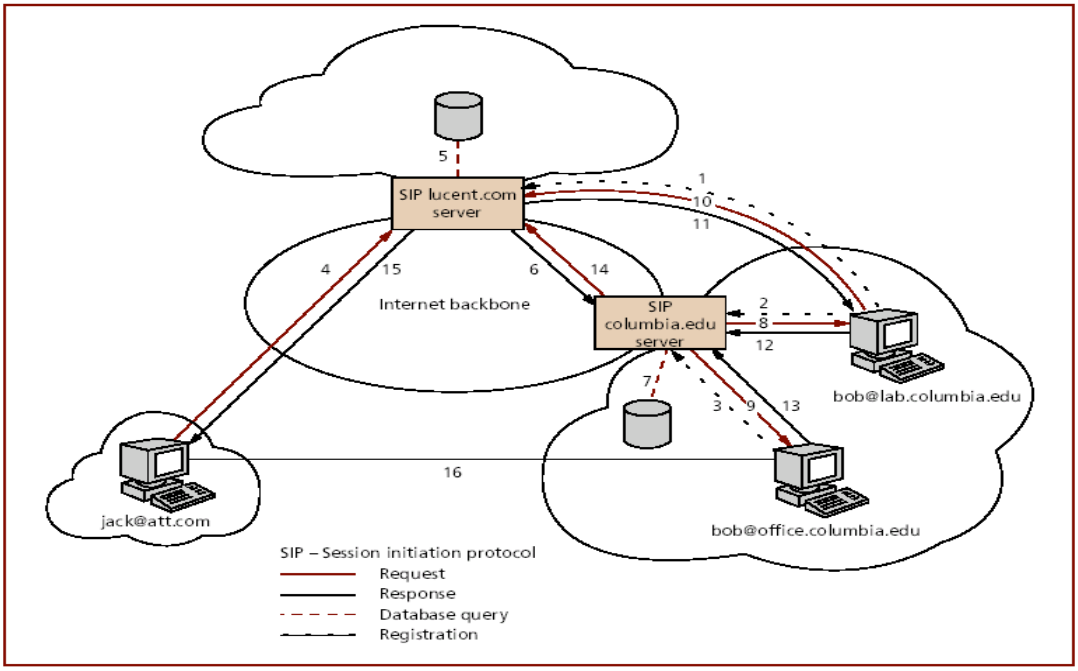

Figure 2.2 Personal Mobility [17]

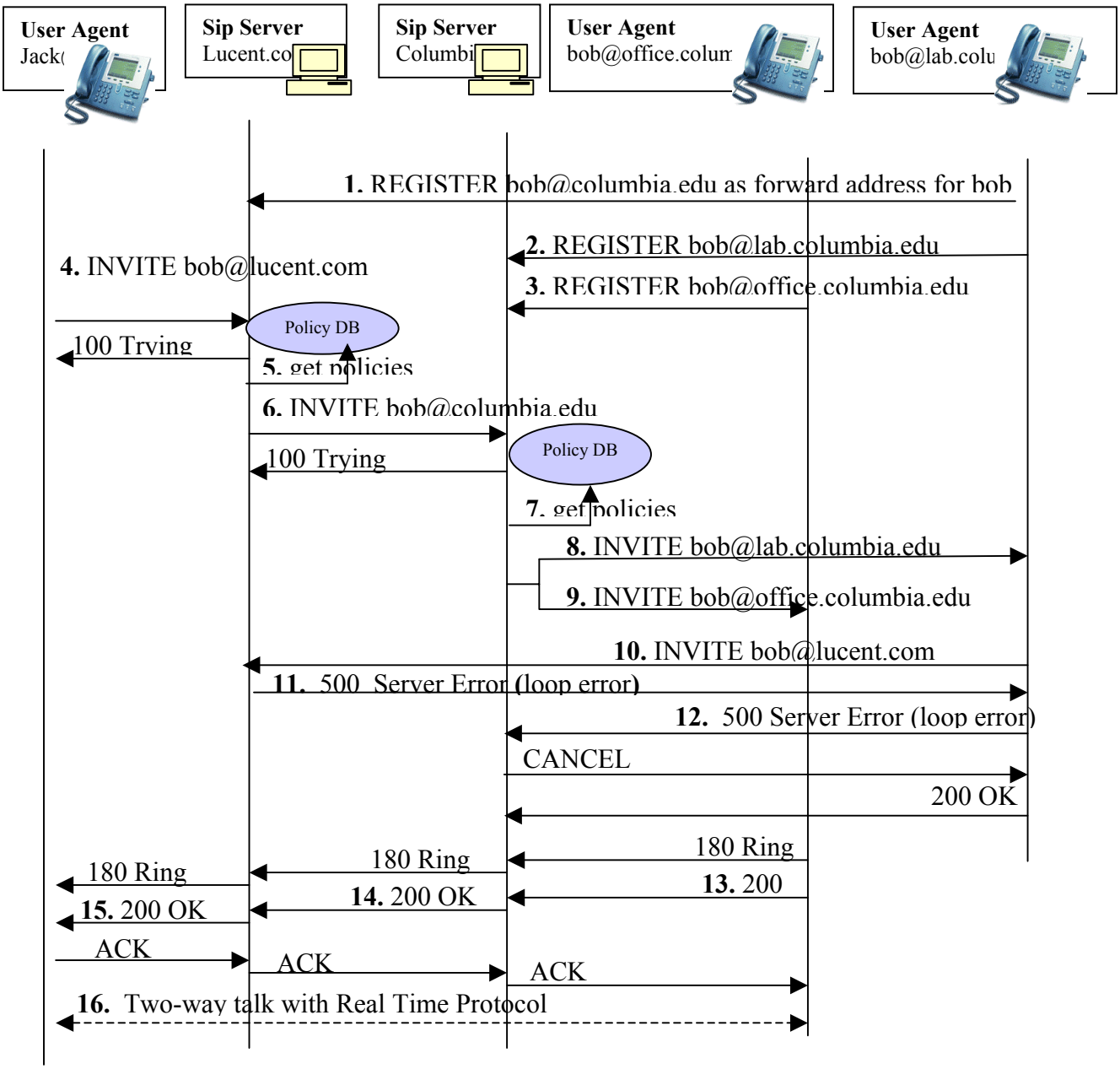

Figure 2.3 MSC of SIP Personal Mobility

Description:

(In this description, the numbers in parentheses indicate the steps in Fig. 2.2)

1. Bob leaves Lucent to Columbia. When he arrives at Columbia University, he registers to Lucent SIP server with his Columbia address: *bob@columbia.edu* as a forwarding address (1).

2. He registers the lab machine - *bob@lab.columbia.edu* and the office machine - *bob@office.columbia.edu* to the Columbia SIP server (2, 3). He previously set his lab's computer to forward calls to his Lucent address, *bob@lucent.com,* but he forgets this. We will see later that this errorous setting will introduce an error message and will be detected. But this will not influence the proper function of forwarding.

3. When bob is at his office in Columbia, Jack initializes a call placed to Bob's public address *bob@lucent.com* at Lucent Technologies location (4).

4. The Lucent SIP server checks its registration and policy in the database and decides to forward the request to *bob@columbia.edu*. By looking up columbia.edu in the Domain Name Server (**DNS**) and the main Columbia SIP server address is obtained and the request is forwarded to it (5, 6).

5. The Columbia SIP server looks up *Bob@columbia.edu* in the database and finds out two end devices listed under the address (7), so it forks and sends a call request to lab and office machine (8, 9) to cause them to ring. This is called **call forking**.

6. The lab phone agent sends the request to the Lucent SIP server by its previous configuration (10). Using the **loop detection** capability in SIP, the Lucent server detects the loop error occurred and sends an error response back to the lab machine (11). In turn, it returns an error code to the Columbia server (12).

7. Bob answers the phone call in his office. The machine sends an acceptance back to the Columbia SIP server (13). After having received both responses back, the SIP server forwards the call acceptance back to the Lucent SIP server (14).

8. The Columbia SIP server forwards the request back to the original caller, Jack (15). All the SIP session states in both Lucent SIP server and Columbia SIP server can be destroyed now.

9. The call is setup and processed by Internet intermediate servers between Jack and Bob (16). Jack and Bob talk on the phone.

This example illustrates three interesting features of SIP. First, it shows that with only one SIP address published, through proper registration and configuration, the network can automatically try all possible locations to reach a user. This is the idea of Personal Mobility we discussed before. Second, the SIP call forking feature allows a call to be forwarded to multiple end devices if those devices are registered under the callee's phone address in a SIP server. Third, the SIP loop detection feature is demonstrated in the Description of step 6. This is really hard to implement in PSTN. In this case, it does not matter if Bob pick up one phone or not. With the SIP loop detection feature, any loops involved in a call can be detected and reported to the corresponding SIP server.

To describe and control additional Internet Telephony services for end users, CPL can be used because CPL works on top of SIP (see Fig 2.1). SIP INVITE messages only deliver incoming or outgoing call requests. CPL scripts are responsible for processing the calls initiated in SIP INVITE messages. CPL services will be introduced in section 2.4.

2.3 End User Services Specified in CPL

2.3.1 Introduction to CPL

Internet Telephony is rich in end user services. In order to describe and control end user services in Internet Telephony, the Call Processing Language (CPL) was developed. The Framework and Requirements of CPL, "RFC2824", was issued in May 2000 [18]. J. Lennox and H. Schulzrinne wrote current CPL specification in January 2002 [9].

CPL [9] was designed for end users to create and control services in Internet Telephony. It works on top of SIP or H.323 and it is very safe for non-professional users because it does not have variables and it can only access limited resources. CPL uses the syntax of the Extensible Markup Language (XML) [10] and this makes CPL simple and easy to extend. CPL allows end users to create and update their own services constantly. It is a programming language in itself as will be explained later.

2.3.2 CPL Structure

A CPL script represents a tree of decisions. In CPL terms, this is represented in terms of tags of nodes and links. Each node or link corresponds to a tag in CPL. A node specifies an action to take or a decision to make. A link specifies the result of an action and displays which decision was made.

CPL Structure in XML version	Explanations
<?xml version="1.0" ?> <!DOCTYPE **cpl** PUBLIC "-//IETF//DTD RFCxxxx CPL 1.0//EN" "cpl.dtd"> <subaction id="voicemail"> services related to voicemails </subaction> <subaction id="email"> services related to emails </subaction> <incoming> services dealing with incoming calls </incoming> <outgoing> services dealing with outgoing calls </outgoing> </cpl>	Specify XML version Specify DTD to be used for validation < -- The script can have zero or more nodes <subaction> with different id specifying how to deal with voicemail or email, zero or one node <incoming> specifying how to deal with incoming calls, zero or one node <outgoing> specifying how to deal with outgoing calls. !-- >

Table 2.4 CPL Structure

There are three types of tags at the highest-level: “subaction”, “incoming” and “outgoing” that each represents a tree itself. The “incoming” tree is executed for incoming calls and the “outgoing” tree is executed for outgoing calls. “subaction” tree can be called by either the “incoming” tree or the “outgoing” tree. They are placed at the root of the CPL script so that they can be called from inside the incoming or outgoing tree. The corresponding tree structure in the XML version is shown in Table 2.4.

CPL processing begins at the incoming or the outgoing tag, the server traverses tree, making decisions or performing actions. There is an implicit global variable in CPL, which is the list of locations. Tags “proxy” and “redirect” use the location list. Tags “location”, “lookup” and location-filter can modify the location list.

SIP Registration	Explanations
REGISTER sip:sip.example.com SIP/2.0 From: Sharon User <sip:sharone@example.com> To: "S. User" <sip:sharone@example.com> CSeq: 18 REGISTER Expires: 1800 Call-ID: 39485832@sharonspc.example.com Contact: sip:sharon@sharonspc.example.com Accept: application/cpl+xml Authorization: Basic am9lOnBhc3N3b3JkAFBX Content-Type: application/cpl+xml Content-Length: 137 Content-Disposition: script Content-Action: add <cpl> <incoming> <address-switch field="origin" subfield="user"> <address is="anonymous"> <reject/> </address> </address-switch> </incoming> </cpl>	Register to the SIP server, "sip: sip.emaple.com" for user Sharon. Sharon has the published SIP address, "sip:sharone@example.com". Sharon contact address is "sip:sharon@sharonspc.example.com". The registration includes the following CPL script. The CPL script is included in the SIP Registration method. Anonymous calls are rejected (see Table 2.7).

Table 2.5 Uploading CPL Scripts via SIP Registration Method

CPL is designed for end users to control their Internet Telephony services. Beside end users, administrator and third parties can also write CPL scripts for services. These services written in CPL can reside on a SIP server, an application server or an intelligent agent. The CPL scripts can be uploaded with a SIP REGISTER (see Table 2.5). CPL scripts can also be uploaded to a server with the aid of end user graphic tools. A user graphic tool is implemented and described in chapter 5 in this thesis. Services written in CPL are suited for call routing services, call screening services, call logging services etc.

When a SIP INVITE message arrives, the CPL script governs its processing if it matches the script. The incoming or outgoing part of the CPL script is invoked, depending on whether it is an incoming or outgoing INVITE. CPL execution terminates when final

action is taken in the CPL script. Since CPL scripts only react INVITE messages, CPL can only be used for pre-call services like routing and screening.

2.3.3 CPL Available Tags

Table 2.6 lists current available tags in CPL. Definitions of these tags can be found in the reference paper [9].

CPL Available Tags	Explanations
Decisions	**Decisions based on**
address-switch	the addresses present in the original call request
string-switch	free-form strings present in a call request
time-switch	the time and/or date the script is being executed
priority-switch	the priority specified for the original call
language-switch	the languages in which caller wishes to communicate
Signaling actions	**Cause signaling events on SIP server**
proxy	forward a call to the current location list
redirect	direct the caller to try the call to the current location list
reject	reject the call attempt
Other actions	**Cause non-signaling events on SIP server**
mail	notify a user of the CPL script status through electronic mail
log	log call information to non-volatile storage
Management tags	**Management actions**
incoming	action triggered by SIP INVITE for incoming calls
outgoing	action triggered by SIP INVITE for outgoing calls
subaction	subaction definitions
sub	subaction references
Location modifiers	**Explicit locations**
location	add literally-specified locations to the current location list
lookup	obtain locations from outside sources
remove-location	remove locations from the location list

Table 2.6 Current Available Tags in CPL

2.3.4 End User Services in CPL

End users can have many kinds of pre-call services. We give some examples to show how to use CPL to describe these services.

Screening services:

As mentioned, current CPL defines five switches. They are address-switch, string-switch, time-switch, priority-switch and language-switch. End users can have screening services based on any of the above switches or any of their combinations. As an example, an end user can reject calls from anonymous callers. This service is based on caller's addresses.

The screening service in CPL is shown in Table 2.7. In the service, incoming calls will be rejected if the user' name is unavailable in the caller's addresses.

Screening Service in CPL	Explanations
<?**xml** version="1.0" ?> <!DOCTYPE **cpl** PUBLIC "-//IETF//DTD RFCxxxx CPL 1.0//EN" "cpl.dtd">	xml declaration cpl definition declaration: cpl version 1.0 at the access URI
<cpl> **<incoming>**	tag <cpl>
<address-switch field="origin" subfield="user">	an address based decision: check the user in the original address
<address is="anonymous">	if the caller is anonymous with unavailable name
<reject/>	the action "reject" is taken and the script stops
</address> </address-switch> </incoming> </cpl>	tag <address> closed tag <address-switch> closed tag <incoming> closed tag <cpl> closed

Table 2.7 Screening Incoming-calls based on Address

Forwarding services:

End users can have forwarding services based on time, address, language, priority or any combination of these. An end user can forward incoming calls to different locations according to his schedule. The service of routine forwarding in CPL is shown in Table 2.8. In the service, user Bob deals with his incoming calls differently by time. He forwards his incoming calls to his office phone in work hours and to his voice mail outside of work hours.

Screening Service in CPL	**Explanations**
<?**xml** version="1.0" ?> <!DOCTYPE **cpl** PUBLIC "-//IETF//DTD RFCxxxx CPL 1.0//EN" "cpl.dtd">	xml declaration cpl declaration: cpl version 1.0 accessible at the specified URI
<**cpl**> <**incoming**> <**time-switch** tzid="America/New_York" tzurl="http://example.com/tz/America/New_York">	cpl contents start deal with incoming-calls check the current time in the time zone, "New-York" at the accessible URI
<**time** dtstart="20000703T090000" duration="PT8H" freq="weekly" byday="MO,TU,WE,TH,FR">	if the time is in work hours (8 hours starting at 9:00am, Monday to Friday every week)
<**proxy**/>	proxy the call to the default address, which is the callee (the user)
</time>	tag <time> closed
<otherwise> **<location** url="sip:bob@voicemail.example.com">	if time condition above is not satisfied, assign the specified url to the location list
<proxy/> </location> </otherwise> </time-switch> </incoming> </cpl>	proxy the call to the location list tag <location> closed tag <otherwise> closed tag <time-switch> closed tag <incoming> closed tag <cpl> closed

Table 2.8 Time-of-day Routing

2.4 Service Mobility through CPL

Service Mobility is harder to achieve. It must be based on Terminal Mobility and/or Personal Mobility. The difficulties are due to the fact that it requires that the service definition be device independent and standardized. As well, a subset of the content of services the user has defined or is subscribed to at his/her home servers must be somehow copied to the servers or end devices that are currently used by the user.

In the example of section 2.3.5, Bob works at Lucent and has an office at Lucent Technologies. He has published a single SIP address, *bob@lucent.com*, which is registered in the Lucent SIP server. Bob is also an assistant professor at the University of Columbia, where he has a lab and an office. Bob has call processing services as described in Fig. 2.2. These services are written in CPL and have been uploaded into the Lucent SIP server in some way.

When Bob works at Lucent, and when his incoming calls arrive at the Lucent SIP server, the SIP server executes CPL scripts: anonymous incoming calls will be rejected; other calls will be forwarded to his office on work hours and to his voice mail out of work hours.

After Bob has arrives at Columbia University, he registers to the Lucent SIP server with his Columbia address: *bob@columbia.edu* as forwarding addresses and registers his office machine -- *bob@office.columbia.edu* to the Columbia SIP server.

When a call is placed to bob's public address *bob@lucent.com* at the Lucent Technologies location, the Lucent SIP server checks its registration and policy in the database and decides to forward the request with a copy of his policies (services in CPL) to *bob@columbia.edu*. By looking up columbia.edu in the Domain Name Server (DNS) to obtain the address of the main Columbia SIP server, the request and the copy of bob's policies are forwarded to the address. Columbia SIP server receives bob's call processing policies and executes the policies: anonymous incoming calls will be rejected; other calls

will be forwarded to his office on work hours and to his voice mail out work hours. The office phone address is found by looking up the database of the Columbia SIP server.

Therefore, end users can have service mobility through the mobility of end user's policies written in CPL. If services are related to time, and the user's current location and original location are in different time zones, the end user can get his original policies, correct the time in the policies and re-upload them to his original SIP server, where his single published SIP address, *bob@lucent.com*, is registered. Of course, some tools will be provided to do time change automatically for the user in the future. In this way, Bob can get exactly the same services at different locations.

2.5 CPL Limitations

As mentioned, current CPL only deals with call handling upon arrival of a SIP INVITE messages. Also, it does not consider any integrated telephony services concerning web, email, instant messaging, presence, text chat, interactive games etc. These services will be considered in CPL extensions. CPL and CPL extensions together can describe many more services for end users in Internet Telephony. Some of these extensions will be discussed in this thesis.

2.6 Conclusion

This chapter has introduced basic concepts of SIP and CPL. Personal mobility, a major feature of SIP, has been described. Services described in CPL have been illustrated. As well, service mobility via CPL has been introduced. The limitations of CPL have been discussed at last.

Chapter 3 Presence System

3.1 Introduction

Presence, also known as presence information, conveys the willingness and the ability of a user to communicate across a set of devices with other users on a network [14]. RFC 2778 [15] defines a model and terminology for describing systems that provide presence information. In the model, a presence system is a presence service that accepts, stores, and delivers presence information to the interested parties i.e. watchers or subscribers. Session Initiation Protocol (SIP) [13] is chosen as the presence protocol when a presence service is provided over Internet. By using SIP, the presence service is compliant with other Internet telephony services because SIP is particularly well suited as a presence protocol and makes the presence service global and reusable.

Note that the services described in this chapter have been discussed in principle in [14]. The architectural model including the three-layer model and the exact message exchange needed to realize these services, are described here for the first time. End user's presence services and system basic services are clearly separated in the thesis.

In this chapter, section 3.2 introduces basic concepts of presence systems in a three-layer architecture. Section 3.3 details the presence system service scenarios. The use of SIP in presence systems is described in section 3.4. Section 3.2 and 3.3 describe our own work, while section 3.4 describes the existing SIP extensions for presence. Section 3.5 discusses the limitations of basic services in presence systems. Section 3.6 is the conclusion of this chapter. This chapter lays the groundwork for presence in SIP and the next chapter will show how extended CPL can be integrated to SIP.

3.2 Concepts

A presence system has two main entities, a presentity and a watcher. A **presentity,** also named a **notifier,** is the logical entity that projects its presence information to the interested parties i.e. the watchers. A **watcher,** also named a **subscriber,** is the logical entity that sends subscription requests to the presentity and receives presence information from the presentity. An end user can be a presentity to his watchers and a watcher to his presentities at the same time. The presentity accepts, stores, and delivers presence information to the interested watchers. Both a presentity and a watcher have a key component, the **Presence Agent (PA)**, which is responsible for sending or receiving subscription requests and notification responses when a watcher and a presentity communicate with each other. Fig. 3.1 displays the presence system architecture in a three-layer model. **SIP services** reside in the lowest layer (i.e. Layer 1). SIP servers provide SIP services to support presence services. **Presence services** include **basic system services** (i.e. the system default services) and **end user specific services**. The basic services reside in the second layer (i.e. Layer 2). The PA, assisted by the presence user agent (PUA), works with the SIP server to provide the presence system services. The end user specific services are supported by the basic services. They reside in the third layer (i.e. Layer 3).

The architecture shown in Fig. 3.1 is described in its functional components and physical components. A physical component can include different functional components. A physical component itself has no functionalities. We will first review the functional architecture of the system and then review its physical architecture.

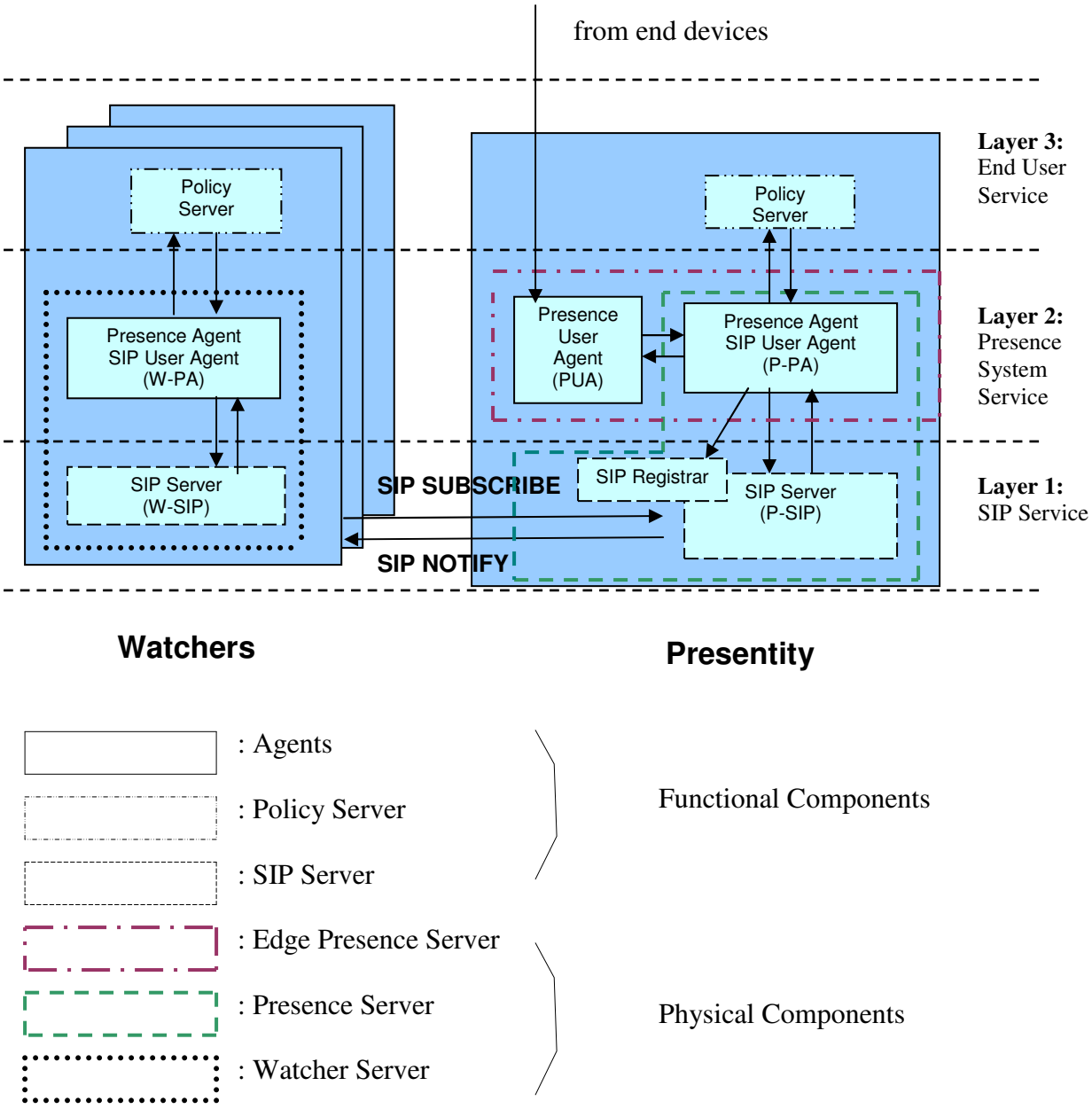

Figure 3.1 Presence System Architecture in a three-layer Model

3.2.1 Functional Architecture

Presence User Agent (PUA):

A Presence User Agent (PUA) is an agent working together with a presence agent for a presentity. A PUA manipulates presence information and pushes the presence data provided by end devices into the presence system for its presentity.

For example, if a presentity has a phone, the PUA takes care of the phone. Upon the request of its presence agent, the PUA gets the phone status information and sends it to its presence agent.

A PUA works for only one presentity, however, a presentity can have multiple PUAs to take care of more than one device such as a phone, cell phone or Personal Digital Assistant (PDA), each of which independently generates a component of the overall presence information for the presentity. The PUAs push data into the presence system, but the PUAs themselves are outside of the presence system. They do not send or receive SIP SUBSCRIBE or SIP NOTIFY messages [14].

Presence Agent (PA):

A presence agent (PA) is a SIP user agent that is responsible for sending and receiving SUBSCRIBE requests; generating, sending and receiving NOTIFICATION responses. The PA is a key component in a presence system.

A **presentity side presence agent** (**P-PA**) has knowledge of the presence state of a presentity. It can generate notifications and can also access presence data manipulated by PUAs for the presentity.

A PA is always addressable with a SIP Uniform Resource Identifier (**URI**) [19] that uniquely identifies the presentity. For an example, a P-PA SIP URI**, "sip:dongmei@site.uottawa.ca"** identifies presentity "dongmei" at the domain "site.uottawa.ca" in the "sip" protocol. A PA uses the SIP REGISTER method to register

the address listed in the "To" header (as shown in Table 2.5 in chapter 2) to its SIP server. In this way, A SIP server identifies the P-PA's current communication addresses.

A P-PA works for one presentity only; however, a presentity can have multiple P-PAs with each of them handling some subsets of all active subscriptions for the presentity. For the case of multiple P-PAs, a P-PA manager is needed to manage these P-PAs. For simplicity, this thesis concerns the case of a presentity with only one P-PA. A P-PA can access a resource containing all of its watchers' URIs. A watcher side presence agent (**W-PA**) can access a resource containing all of its presentities' URIs.

A P-PA also works for the presentity as a notifier [20]. It supports the presence event package that is delivered in SIP NOTIFY messages. This will be discussed in detail in section 3.3.

3.2.2 Physical Architecture

Watcher Server:
A watcher server is a physical entity on the watcher side. It can act as either a presence agent or as a SIP server for SUBSCRIBE requests and NOTIFY responses [14].

When the watcher server acts as a presence agent, it is aware of the acceptable format of the presence document. When the presence server acts as a SIP server, the SUBSCRIBE requests are proxied to the P-PAs of its presentities and the NOTIFY responses are received from its presentities.

According to Fig. 3.1, there are two possible architectures for the presentity [14]. We call them presence server and edge presence server architectures. In the case of Edge presence server, presence data is stored locally in PUA and it can be accessed faster. In the case of presence server architecture, the presence data are centrally stored, controlled and managed in the database of the SIP registrar server.

Presence Server:

A P-PA can co-locate with its SIP proxy/register to form a presence server. The presence server is a physical entity on the presentity side. It can act as either a presence agent or as a SIP server for SUBSCRIBE requests and NOTIFY responses.

When the presence server acts as a presence agent, it is aware of the presence information of the presentity through some non-SIP protocol means. When the presence server acts as a SIP server, the SUBSCRIBE requests are delivered to the P-PA of its presentities and NOTIFY responses are proxied to the W-PAs of its watchers.

If it acts as a presence server, a P-PA gets presence information that is pushed by PUAs and is stored in the registration database. The P-PA co-locating with the proxy/registrar can access the database to get the presence information. The P-PA must be informed of presence info changes in the database.

Edge Presence Server:

Another possibility is that the P-PA co-locates with its PUAs to form an edge presence server. The edge presence server is a physical entity. It is aware of the presence information of the presentity because it co-locates with the entities that manipulate the presence status of its presentity.

If it acts as an edge presence server, a P-PA gets presence information. The PUAs maintain the presence states and report all changes to the P-PA directly. The P-PA can get the presence information and assemble it into a presence document. The document will be delivered in a SIP NOTIFY message, which will be discussed in detail in section 3.3.

SIP Server:

The SIP server works on the first layer to back up the PAs in a presence system. It is a physical entity that provides SIP services, for an example, sending SIP SUBSCRIBE requests and receiving SIP NOTIFY responses on the watcher side; receiving SIP

SUBSCRIBE requests and sending SIP NOTIFY responses on the presentity side. The use of SIP will be described in section 3.4.2. SIP servers physically realize the presence system protocol, SIP.

Policy Server:

The policy server is the server that manages end user specific presence services (i.e. policies). The policy server works on the third layer. The management includes creating, storing, updating, deleting, searching, and fetching presence policies for end users.

3.2.3 SIP Operations for Presence

There are two SIP extension messages for presence i.e. SIP SUBSCRIBE and SIP NOTIFY. A SIP SUBSCRIBE message is a request message to a presentity sent by a watcher. A SIP NOTIFY message is the presentity's final response sent to the watcher. The "final" means that the response contains the presentity's presence information required by the watcher.

SUBSCRIBE Operation:

The SIP SUBSCRIBE operation is a request operation initiated by a W-PA and sent by the **watcher side SIP server** (**W-SIP Server**). In order to get presence information of a presentity, the SUBSCRIBE request is prepared by the W-PA. The W-PA identifies the destination presentity by using a SIP URI. W-SIP server receives the SIP URI and sends a SIP SUBSCRIBE request message to the presentity.

The SIP SUBSCRIBE message contains the SIP URI of the presentity and the required format of presence information. The SIP SUBSCRIBE operation initiates a presence service "dialog" between the W-PA and the P-PA. If the request is approved, the water is registered successfully to the presentity.

NOTIFY Operation:

The SIP NOTIFY operation is a final response operation initialized by a P-PA and sent by **presentity side SIP server** (**P-SIP server**).

After a P-PA gets the SIP SUBSCRIBE message from its P-SIP server, it must authenticate and authorize the subscription request before it sends the final response back to the watcher. This authentication must be done using one of the mechanisms defined in [13]. The authorization is done under the guide of the system and end user policies. This will be described in Section 3.4.

To respond to a watcher request, the P-PA prepares a presence document and identifies the destination watcher using SIP URI. The P-PA sends the document with the watcher's SIP URI to its P-SIP server. After the P-SIP server receives the notification information, it sends a SIP NOTIFY message to the destination watcher. The SIP NOTIFY message contains the SIP URI of the watcher and the presence information of the presentity. This SIP NOTIFY operation is the final response to the watcher's request.

A SIP NOTIFY operation can be induced by any of the following events:

1. A presentity registers its presence information.
2. The presence information is updated.
3. A watcher subscription request is approved.

3.3 Presence System Service Scenarios

As mentioned in section 3.2, presence systems have two types of participants, watchers and presentities. A watcher can send a subscription request to a presentity and get the

presentity's presence information. The presentity can process the watcher's request and notify the watcher of its presence information.

In this section, how the components of the presence system play together to support the presence services will be described in the system service scenarios. There are five main scenarios to illustrate the basic services of presence systems. The parts in italic are related to end user policies that will be discussed in detail in Chapter 4. In this chapter, we will only mention the use of these policies in the system service scenarios. Chapter 4, together with this chapter, will illustrate all the presence services including basic services and user specific services.

There are two different ways for a P-PA to get presence information. In the case of the edge presence server described in section 3.4.2 and section 3.4.4, the P-PA co-locates with PUAs. In the case of the presence server, described in section 3.4.5, the P-PA co-locates with a SIP proxy/registrar.

Note that in some cases, we make distinction between major and minor steps. For example, in Fig. 3.2, Step 1 consists of four sub-steps 1-1 to 1-4. When steps in one scenario follow of steps in another, we continue the number of steps. For example, step 3 in Fig. 3.3 is the same as step 3 in Fig. 3.2.

3.3.1 To Send a Subscription Request

Upon request from the watcher to subscribe to the presentity on its presence information, the W-PA will check the out-going subscription policies and then will ask SIP server to send a SIP SUBSCRIBE to the presentity (see Fig. 3.2).

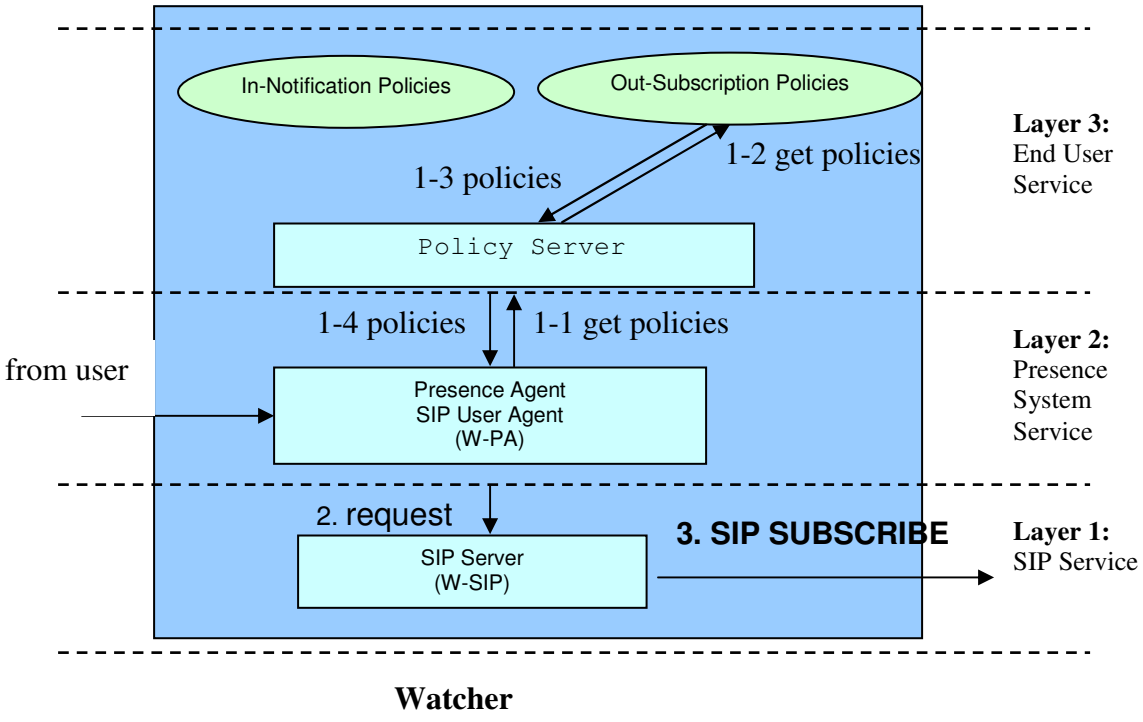

Figure 3.2 A Process on an Outgoing-SUBSCRIBE

Description:

(In this description, the numbers in parenthesis indicate the steps shown in Fig. 3.2)

1. A W-PA prepares a subscription request and sends the request to its W-SIP server. The request includes the SIP URI address of the W-PA, the SIP URI address of the destination P-PA, the expiration time of the request dialog etc. As the system default, the watcher does not have any policies for the outgoing-subscription request, as seen in step (2).

   If the watcher does have its own policies for outgoing subscription requests, the W-PA processes the outgoing request according to the policies i.e. steps (1-1), (1-2), (1-3) and (1-4) are taken before step (2) is taken. Please refer to section 4.5.2.

2. After the W-SIP server receives the request from its W-PA, it sends a SIP SIBSCRIBE message to the destination P-PA that is identified by its SIP URI, as seen in step (3).

3.3.2 To Process a Watcher's Request

In the case of a watcher's request, the P-PA may have to approve it, subject to the incoming-subscription policies. If the request is approved, then the P-PA will ask the PUA for presence information, and will send it back with the SIP NOTIFY message, subject to outgoing-notification policies (see Fig. 3.3).

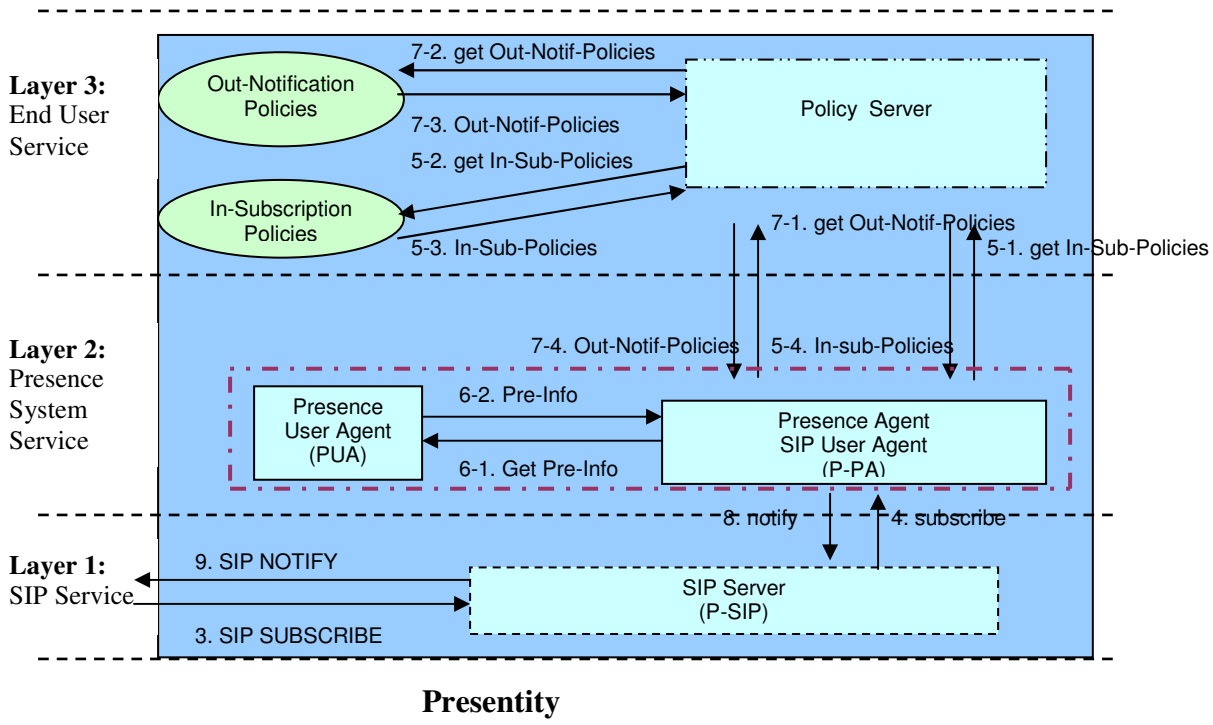

Figure 3.3 A Process on an Incoming-SUBSCRIBE and Outgoing-NOTIFY

Description:

(In this description, the numbers in parenthesis indicate the steps in Fig. 3.3)

1. After a P-SIP server receives a watcher's subscription request, the P-SIP server disposes of the request information and passes it to the P-PA of the presentity, as seen in step (4).

2. The P-PA authenticates the watcher after it receives the resolved request information. In the case of system default, the presentity itself does not have any authorization policies for requests. The watcher is approved.

   If the presentity does have his own authorization policies for incoming subscription requests, the P-PA processes the request according to the policies, as shown in steps (5-1), (5-2), (5-3) and (5-4). Please refer to section 4.5.3.

3. If the subscription request is approved, the P-PA fetches presence information from its PUA, see steps (6-1) and (6-2). In the case of system default, the presentity itself does not have any policies for the outgoing-notification responses. The P-PA writes a presence document in Presence Information Data Format (**PIDF**) [21]. The PIDF is the required format described in the previously received SIP SUBSCRIBE message. The P-PA sends the presence document together with the SIP URI of the request W-PA to its P-SIP server, as seen in step (8).

   If the P-PA cannot get the presence information because of some reasons, such as that the presentity has not registered its presence information or the subscription is not approved, the P-PA wraps the information of notice to a blank presence document and sends the document with the SIP URI of the request W-PA to its P-SIP server, as seen in step (8).

   If the presentity does have policies for outgoing-notifications, the P-PA processes the outgoing response according to the policies, i.e. steps (7-1), (7-2), (7-3) and (7-4) are taken before step (8) is taken. If the watcher's request is rejected, the P-

PA wraps the information of rejection to a blank presence document and sends the document with the SIP URI of the request W-PA to its P-SIP server, as shown in step (8). Please refer to section 4.5.4.

4. After the P-SIP server receives presence information, it sends a SIP NOTIFY message to the requesting W-PA identified by its SIP URI. The SIP URI of the W-PA is written into the "To" field and the SIP URI of P-PA is written into the "From" field in the SIP NOTIFY message in step (9).

3.3.3 To Receive a Notification Response

The W-SIP server receives the SIP NOTIFY and passes it to the W-PA, The W-PA will provide service subject to incoming-notification policies (see Fig. 3.4).

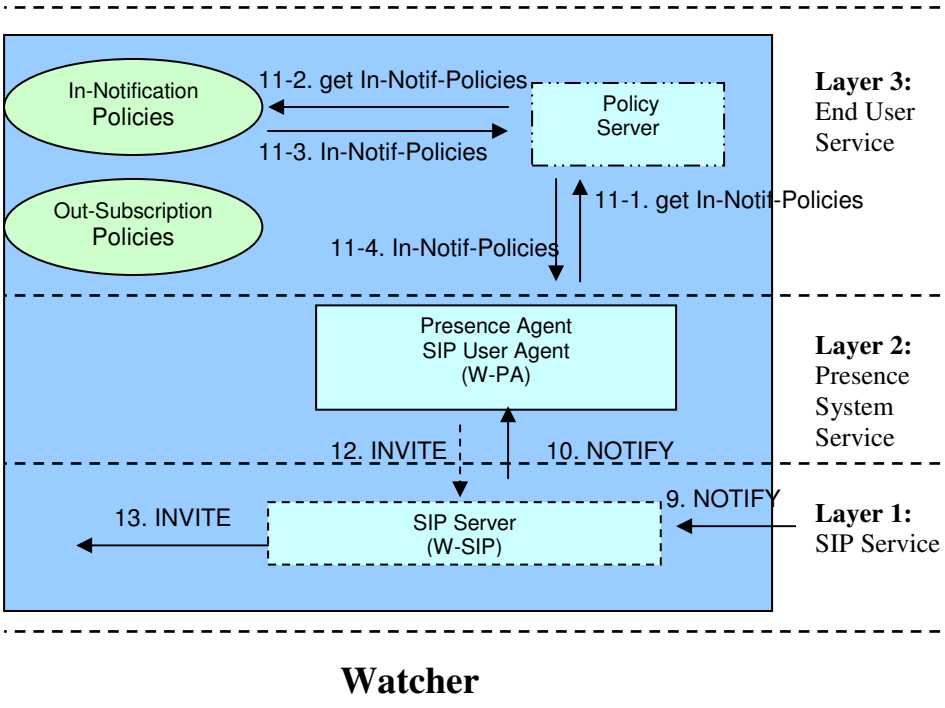

Figure 3.4 A Process on an Incoming-NOTIFY

Description:

(In this description, the numbers in parenthesis indicate the steps in Fig. 3.4)

1. After a W-SIP server receives a SIP NOTIFY message sent by its P-SIP server, as seen in step (9), the W-SIP server resolves the SIP NOTIFY message and passes it to the W-PA, as shown in step (10).

2. In the case of system default, the watcher does not have any policies for incoming notifications. The W-PA receives the notification and displays the presence information of the presentity.

   If the watcher does have his policies for incoming notifications, the W-PA processes the notification message according to the policies, i.e. steps (11-1), (11-2), (11-3) and (11-4) are taken. Please refer to section 4.5.5.

3.3.4 To Register or Update Presence Information (edge device server case)

The following procedure is executed when the presentity registers to the system or the end devices detect a change in user status. This could be a change of location, a change in availability of user or an action such as picking up the phone. The P-PA will be informed that the change has occurred and then can ask to update the presence information. The P-PA will then notify its registered watchers, subject to its policies (see Fig. 3.5).

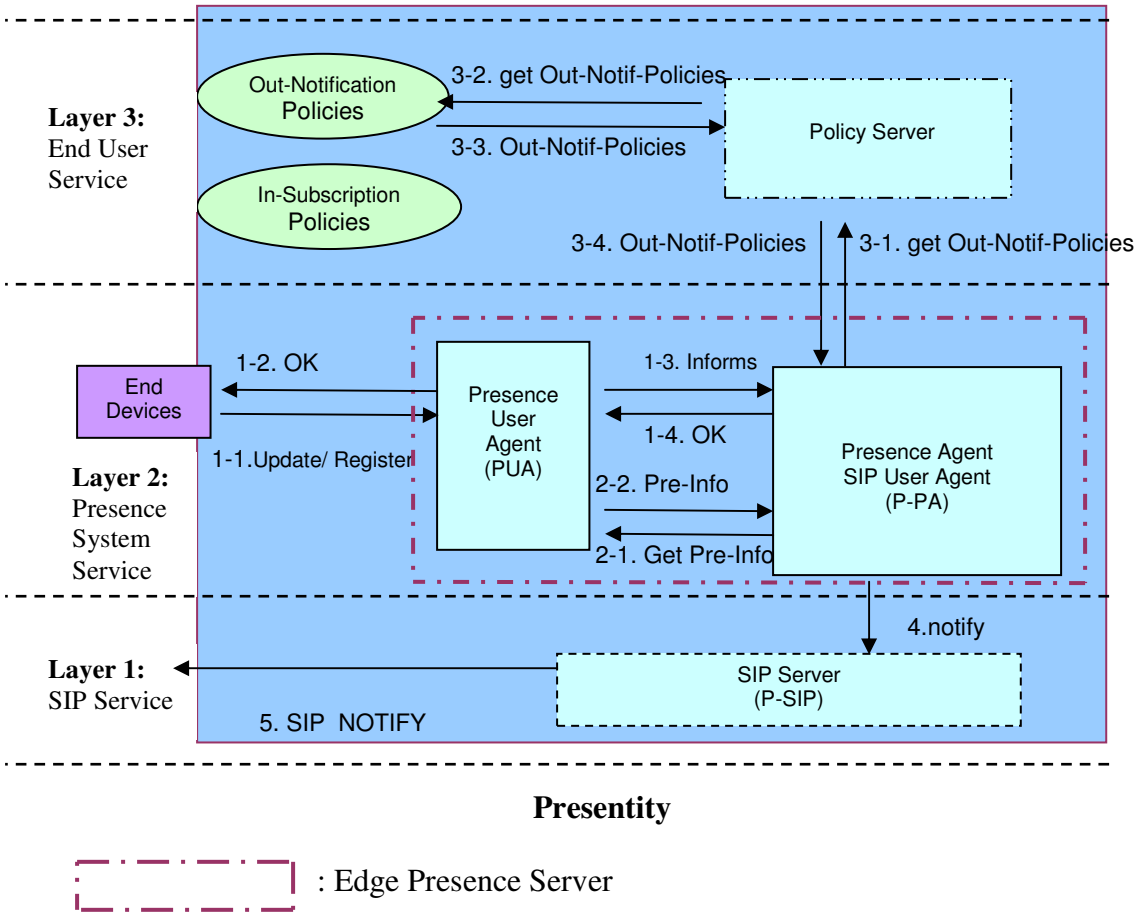

Figure 3.5 A Process on Registration or Updating (edge device server case)

Description:

(In this description, the numbers in parenthesis indicate the steps in Fig. 3.5)

1. When an end device of a presentity registers or updates presence status to its PUA, as seen in steps (1-1) and (1-2), the PUA records the presence information and informs its P-PA, as seen in steps (1-3) and (1-4).

2. After the P-PA receives the notice, it fetches presence information from its PUA, as shown in steps (2-1) and (2-2) and prepares notification information. We assume that the presentity itself does not have any policies for the outgoing notification responses. The P-PA writes a presence document for its presentity in PIDF [21]. The PIDF is the required format in the previously received SIP SUBSCRIBE request message. The P-PA sends the document together with the SIP URI of the request W-PA to its P-SIP server, as seen in step (4).

   If the presentity does have policies for outgoing-notifications, the P-PA deals with the outgoing responses according to the policies i.e. steps (3-1), (3-2), (3-3) and (3-4) are taken before step (4) is taken.

3. After the P-SIP server receives presence information, it sends a SIP NOTIFY message to the request W-PA, as seen in step (10).

3.3.5 To Register or Update Presence Information (presence server case)

This case is the same as the one in the previous section. However, a different architecture is assumed. Instead of having an edge presence server, we have a presence server (see Fig. 3.6).

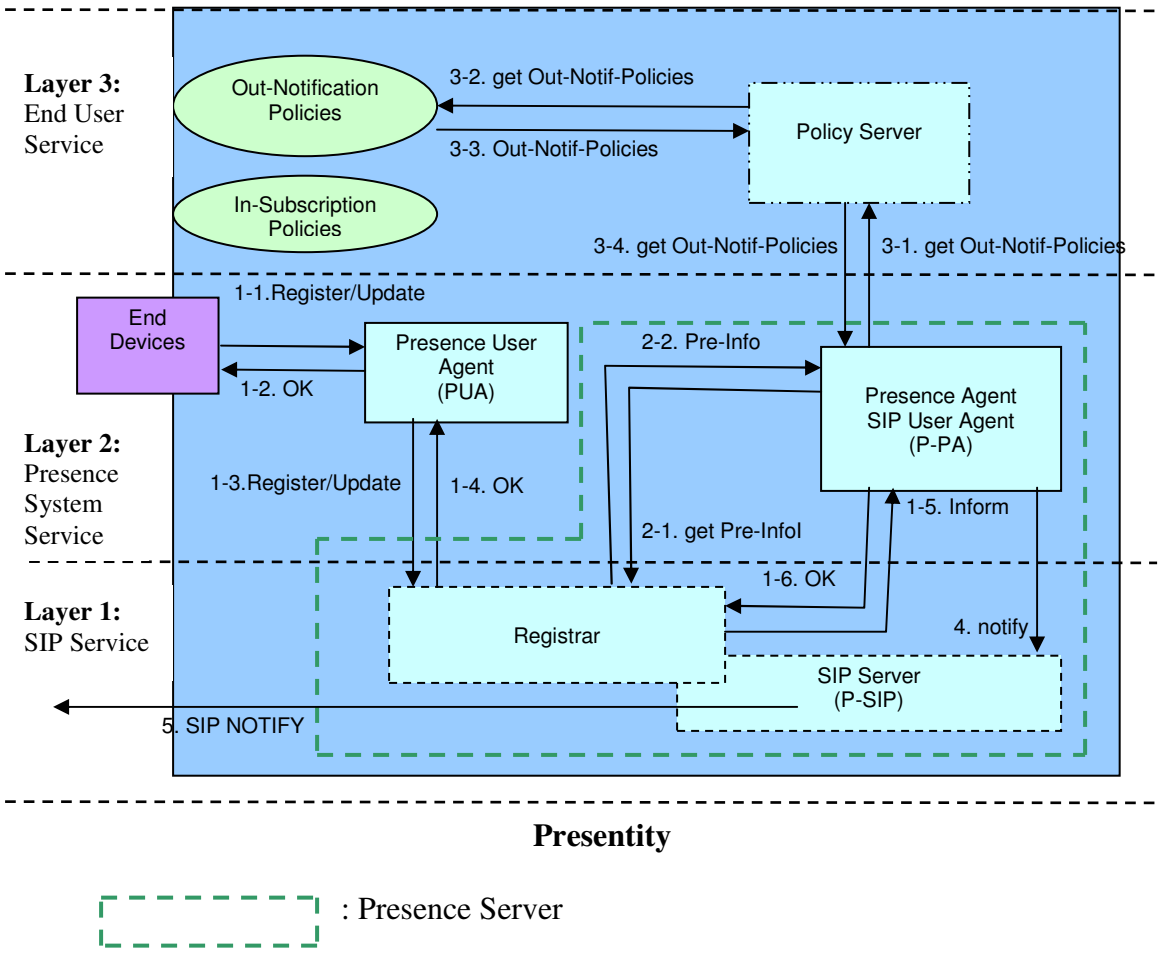

Figure 3.6 A Process on Registration or Updating (presence server case)

Description:

(In this description, the number in parenthesis indicates the steps in Fig. 3.6)

1. End devices of a presentity register or update the presence information in the PUA, as shown in steps (1-1), (1-2). The PUA registers or updates presence information to the presence registrar for its presentity, as seen steps (1-3) and (1-4). The registrar saves the presence information in its database and informs the P-PA of the PUA, as seen in steps (1-5) and (1-6).

2. After the P-PA is informed, the P-PA fetches presence information from the database in the registrar, as seen in steps (2-1) and (2-2), and prepares a notification message. We assume that the presentity itself does not have any policies for the outgoing-notification response. The P-PA writes a presence document in PIDF [21]. The PIDF is the required format in the previously received SIP SUBSCRIBE messages sent by its watchers. The P-PA sends the document together with the SIP URI of the request W-PA to its SIP server, as shown in step (4).

   If the presentity does have policies for the outgoing notification, the P-PA needs to process the outgoing responses according to the policies, i.e. steps (5), (6), (7) and (8) are taken before step (4) is taken.

3. After the P-SIP server receives presence information, it sends a SIP NOTIFY message to the request W-PA, as shown step (5).

3.4 The Use of SIP in Presence Systems

The Session Initiation Protocol (SIP) plays the role of the presence system protocol, which is a very important role in the physical architecture. SIP is an application-layer control protocol that can establish, modify and terminate presence service dialog sessions in presence systems.

3.4.1 Why Choose SIP as the Presence System Protocol

SIP is used as the presence system protocol when a presence service is provided over the Internet. There are at least two main reasons for choosing SIP as the presence protocol. First, presence as one of the Internet telephony services is compliant with other Internet services if the SIP presence protocol is used. Secondly, the use of SIP offers presence users access to the SIP network. The SIP network allows a watcher's subscription request

to be routed to the server on which the watcher's registration is hosted. This SIP connectivity makes the presence service global and reusable.

3.4.2 SIP Extensions for Presence

SIP extensions for presence include two SIP messages, i.e. **SIP SUBSCRIBE** and **SIP NOTIFY** [14]. Watchers' requests and presentities' responses are physically delivered in the two types of messages, the SIP SUBSCRIBE messages and SIP NOTIFY messages.

The SIP message flow, as shown in Fig. 3.7, describes message passing between a W-SIP server and a P-SIP server. It is assumed that the watcher has previously been authorized to subscribe to this presentity resource at the presence server and that the PUA is responsible for informing the presence server of the change of presence status through some non-SIP means.

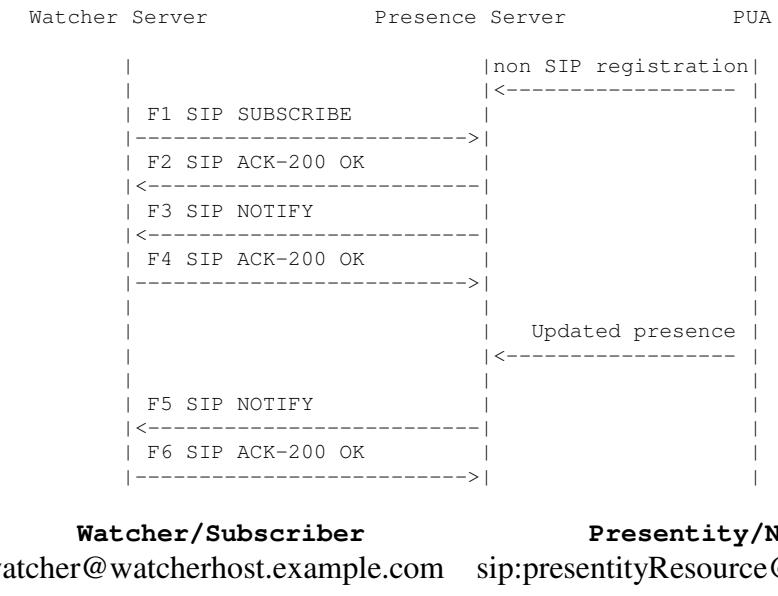

Figure 3.7 A SIP Message Flow in a Presence System [14]

The presence server can access presence information after the PUA registers the presence information for its presentity.

F1 SIP SUBSCRIBE:

The SIP SUBSCRIBE message describes a watcher's subscription request to its presentities. It contains the SIP URI of its destination P-PA, its expiration time and the acceptable data format of the presentity's presence document.

The SIP SUBSCRIBE message is initialized by the W-PA and sent by the W-SIP server. The W-PA is identified by "sip:wtcher@watcherhost.example". The P-PA is identified by "sip:presentityResource@example.com" (see Fig. 3.1).

F1: SIP SUBSCRIBE Message	**Explanations**
SUBSCRIBE sip:presenceResource@example.com SIP/2.0	sip version2.0
Via:SIP/2.0/TCP watcherhost.example.com;branch=z9hG4bKnashds7	
To: **<sip:presentityResource@example.com>**	Presentity SIP URI
From: **<sip:watcher@example.com>**;tag=xfg9	Watcher SIP URI
Call-ID: 2010@watcherhost.example.com	
CSeq: **17766 SUBSCRIBE**	
Max-Forwards: 70	
Event: presence	presence event package
Accept: application/cpim-pidf+xml	required format of presence doc
Contact: <sip:watcher@watcherhost.example.com>	contact address
Expires: **600**	expiration time
Content-Length: 0	

Table 3.1 F1: SIP SUBSCRIBE Message

F2 SIP 200 OK:

After the presence server receives the SIP SUBSCRIBE request message from the watcher server, the presence server informs the watcher server that the subscription request was successfully received. This F2 200 OK message is a response to F1 SIP SUBSCRIBE, so they have the same consequence number i.e. "17766 SUBSCRIBE". They have the same Call-ID, "`2010@watcherhost.example.com`" because they are in the same conversation dialog (see Fig. 3.2).

F2: SIP 200 OK Message	Explanations
SIP/2.0 200 OK Via: SIP/2.0/TCP watcherhost.example.com;branch=z9hG4bKnashds7 ;received=192.0.2.1 To: **<sip:watcher@example.com>**;tag=xfg9 From: **<sip:presentityResource@example.com>**;tag=ffd2 Call-ID: 2010@watcherhost.example.com CSeq: **17766 SUBSCRIBE** Event: presence Expires: 600 Contact: sip:server.example.com Content-Length: 0	SIP version 2.0 destination SIP URI P-PA SIP URI Subscribe sequence presence event expiration time contact address

Table 3.2 F2: SIP 200 OK Message

F3 SIP NOTIFY:

F3: SIP NOTIFY Message	Explanations
NOTIFY sip:watcher@watcherhost.example.com SIP/2.0 Via: SIP/2.0/TCP server.example.com;branch=z9hG4bKna998sk From: **<sip:presentityResource@example.com>**;tag=ffd2 To: **<sip:watcher@example.com>**;tag=xfg9 Call-ID: 2010@watcherhost.example.com Event: presence Subscription-State: active; expires=**599** Max-Forwards: 70 CSeq: **8775** NOTIFY Contact: sip:server.example.com Content-Type: application/cpim-pidf+xml Content-Length: …… **[PIDF Document in format "application/cpim-pidf+xml"]** ……	SIP version 2.0 P-PA SIP URI Watcher SIP URI Dialog ID presence event Notify sequence 8775 content format presence document

Table 3.3 F3: SIP NOTIFY Message

A SIP NOTIFY message describes the final response to the watcher's subscription request sent by the presence server. This final response message contains presence information of the presentity in the PIDF [21] i.e. "application/cpim-pidf+xml". Call-ID "2010@watcherhost.example.com" shows that the F3 SIP NOTIFY message is in the

same dialog as F1 SIP SUBSCRIBE and F2 SIP 200 OK. The dialog has 599 seconds left to be alive (see Fig. 3.3).

The PIDF, as a common presence data format for Common Profiles for Presence (CPP)-compliant presence protocols, allows presence information to be transferred across CPP-compliant protocol boundaries without modification, with the added benefits of security and performance. The PIDF will be described in detail in chapter 4.

F4 SIP 200 OK:

After the watcher server receives the notification message from the presence server, it sends SIP 200 OK message to inform the presence server that the notification message is received (see Fig. 3.4).

F4: SIP 200 OK Message	**Explanations**
SIP/2.0 200 OK	SIP version 2.0
Via: SIP/2.0/TCP server.example.com;branch=z9hG4bKna998sk ;received=192.0.2.2	
From: **<sip:watcher@example.com>**;tag=xfg9	Watcher SIP URI
To: **<sip:presentityResource@example.com>**;tag=ffd2	Destination SIP URI
Call-ID: 2010@watcherhost.example.com	Dialog ID
CSeq: 8775 NOTIFY	Notify sequence 8775
Content-Length: 0	

Table 3.4 F4: SIP 200 OK Message

F5 SIP NOTIFY:

When the presence server is informed by a PUA that the presence status is updated, it sends the updated presence information to the watchers of the presentity. In this SIP NOTIFY message, the updated presence information is written in the same format, “application/cpim-pidf+xml”, as the previous notification message. This message initiates a new notification sequence identified by number 8776 and the dialog has 543 seconds left to be alive (see Fig. 3.5).

F5: SIP NOTIFY Message	**Explanations**
NOTIFY sip:watcher@watcherhost.example.com SIP/2.0 Via: SIP/2.0/TCP server.example.com;branch=z9hG4bKna998sl From: **<sip:presentityResource@example.com>**;tag=ffd2 To: **<sip:watcher@example.com>**;tag=xfg9 Call-ID: 2010@watcherhost.example.com CSeq: **8776 NOTIFY** Event: presence Subscription-State: active; **expires=543** Max-Forwards: 70 Contact: sip:server.example.com Content-Type: application/cpim-pidf+xml Content-Length: ... **[PIDF Document in format "application/cpim-pidf+xml"]** …….	SIP version 2.0 P-PA SIP URI destination SIP URI Dialog ID Consequence notify 8776 presence event dialog can last 543 seconds contact address content format content

Table 3.5 F5: SIP NOTIFY Message

F6: SIP 200 OK Message	**Explanations**
SIP/2.0 200 OK Via: SIP/2.0/TCP server.example.com;branch=z9hG4bKna998sl ;received=192.0.2.2 From: **<sip:presentityResource@example.com>**;tag=ffd2 To: **<sip:watcher@example.com>**;tag=xfg9 Call-ID: 2010@watcherhost.example.com CSeq: 8776 NOTIFY	SIP version 2.0 P-PA SIP URI Destination SIP URI Dialog ID Notify sequence 8776

Table 3.6 F6: SIP 200 OK Message

F6 SIP 200 OK:

After the watcher server receives the new notification, the watcher informs the presence server by sending a new SIP 200 OK message to it (see Table 3.6).

In this SIP 200 OK message, the consequence number "8776 NOTIFY" illustrates that the F5 200 OK message is in the NOTIFY sequence identified by number "8776".

3.5 Basic Service Limitations

Throughout the illustrations and discussions in this chapter we can see that the SIP presence system offers the same basic services to all its users, which include

1. Store and manage presence information for a presentity.
2. Send a watcher's subscription request.
3. Receive a watcher's subscription request and send a notification back to the watcher.
4. Send notifications to all watchers when presence information is registered.
5. Send notifications to all watchers when presence information is updated.

None of the above services concerns end users' specific needs. However, end users would like to have their specific services. For example, Tom may wish to reject all the watcher requests after working hours, Alice would like the notification to Tom not to occur more than every 10 minutes etc. The basic presence system described in this chapter cannot offer these services. We will extend CPL and presence information to include these new services in Chapter 4.

3.6 Conclusion

The basic concepts of SIP presence systems were introduced in a three-layer architecture in this chapter. The use of SIP in the presence systems was described and scenarios were illustrated. End user's presence services and system basic services were clearly separated. The basic services in presence systems have some limitations. In chapter 4, we will discuss these limitations and will show how additional services can be provided to end users through the extensions of CPL and presence information.

Chapter 4 Extensions of Presence Information and CPL

There are two types of services in a presence system, **system basic services** and **end user services.** The system basic services are independent of physical end users. This means that the presence system provides exactly the same services to all of its users. These basic services have been discussed in chapter 3.

In a presence system, end user specific services are also called **end user policies**. End users can write their own policies in CPL for specific needs. In this chapter, we illustrate how to extend presence information to get rich presence information in the presence services and how to extend CPL to describe user specific presence related services. New services provided through the extensions are illustrated through various examples.

Section 4.1 describes the end user specific services in the three-layer model for presence service. Section 4.2 discusses the reason why Extended CPL is chosen to describe end user services in a presence system. Section 4.3 illustrates how to extend presence information. The CPL extensions for presence are described in section 4.4. User specific presence services specified in extended CPL are illustrated in section 4.5. Section 4.6 shows new call processing services related to presence and Section 4.7 is the conclusion of this chapter. The work presented in this chapter is our own contribution, based on the existing SIP and CPL described in chapter 2 and chapter 3.

4.1 End User Services in a Presence System

End user policies define specific services for physical end users. End user policies reside in the high layer and basic system services reside in the low layer, see the service model in Fig. 4.1. End user policies use the basic system services. End users have the basic system services in common and have their specific services in difference.

The basic system services shown in Fig. 4.1 have been described in chapter 3. These services include: watchers can send subscription requests to presentities; presentities approve these requests and send final notifications back to the watchers.

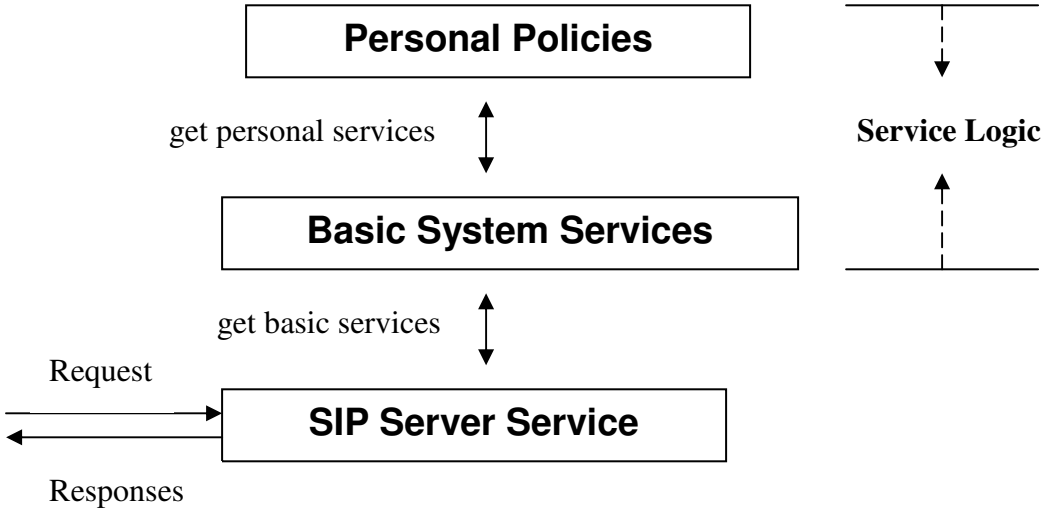

Figure 4.1 Service Model in a Presence System

In the real world, end users need not only the basic services but also their own specific services, as shown in Fig. 4.1. These specific services are desirable for end users but are not included in the presence systems described in chapter 3.

Here we give an example to show what these policies may be. Sharon is a user of a presence system. She works as a consultant for a company and as a professor for a university as well. As an end user of the presence system, Sharon likes to have the following policies:

1. Sharon only approves subscription requests in the time slots 8:30-12:00am and 12:30-5:00pm, Monday to Friday.

2. Sharon rejects all subscription requests written in Japanese.

3. Sharon rejects all subscription requests from her company when she takes the role of “professor”.

4. Sharon rejects all subscription requests from the university when she takes the role of “consultant”.

5. When the presence information is updated, Sharon notifies her watchers differently. Her company watchers are notified when Sharon takes the role of “consultant” and her university watchers are notified when Sharon takes the role of “professor”.

6. When Sharon’s presence information is updated, the frequency of the above notifications should be less than every 5 minutes.

7. When Sharon takes the role of “consultant”, she would like to have an automatic call to the boss of her company as soon as her boss comes to his office.

This kind of services can be applied in some organizations, such as governments, military, universities etc.

4.2 Motives for Choosing CPL Extensions for Presence

We know that CPL is designed for end users to describe their services in Internet telephony. The current CPL specification focuses on call handling. It does not support any presence related services. To support these services, we can either create a new language or extend CPL. Extending CPL is a better choice than creating a new language due to the following three reasons. First, many existing CPL components can be reused. These components are still very useful for end users in a presence system. Second, CPL extensions for presence are not tied to any particular presence protocols so that they are compliant with the Common Profile for Instant Messaging (CPIM) [22] and the presence

event package for SIP [14]. Last, components defined in CPL extensions for presence can be reused in call handing systems. Actually, CPL extensions can be used to describe the combined services of presence systems and call handling systems.

4.3 Extensions of Presence Information

Presence information is the basis of presence related services. In the past, presence has been limited to “on-line” and “off-line” indicators. The notion of presence in this thesis is much broader. It can be a location of a user, the role that a user is currently taking or a user’s willingness to communicate etc. These extensions of presence information will much enrich presence related services.

Users in different organizations may need different presence information. The presence information can be extended or defined by users in XML. There are two primary formats to describe the structure of an XML document. The first one, the Document Type Declaration (DTD), is part of the original XML specification. The second format is the XML schema. Presence documents are in Presence Information Data Format (PIDF) [21] using XML syntax. The extensions of presence information can be defined in either of the two formats.

A presence document written in PIDF declares that the definition format, DTD or XML schema, is used. The format is used when validating the presence document at loading time. The definition format file is identified by a name and a corresponding URI. The format file must be public and accessible through the URI when the presence document needs validation.

As an example of presence extensions, four new presence parameters are introduced in section 4.3.1. Section 4.3.2 defines the extensions of presence information. Section 4.3.3 illustrates a presence document containing extended presence information. The presence

documents in PIDF are included in and delivered with SIP NOTIFY messages to watchers.

4.3.1 Extensions of Presence Information

RFC2779 [23] requires that PIDF have means of extending presence data. These extensions merely allow protocols and applications to define richer presence data. They should not change the structure or semantics of the PIDF bodies.

As an example of the extensions of presence information, four presence parameters are defined as follows.

Location:

Parameter "location" indicates a user's current location. The parameter value can be "office", "meeting room" or "car" etc. These locations must be equipped with sensors in a network. The sensors can identify and monitor the users who have registered in the network.

For an example of location change, if a user i.e. a presentity leaves his office to go to the meeting room, the sensor in his office notices that the user is not in the office and the sensor in the meeting room notices that the user appears in the meeting room. A monitor of the sensor network notices the location change of the user and the monitor can inform the user's P-PA directly. The sensor network works as a PUA and as an end device for the presence system. After the P-PA gets the notice of location change, it notifies all watchers of the user. The system scenario has been described in section 3.3.4.

Phone Line Status:

Parameter "lineStatus" indicates if the user i.e. a presentity is occupying a telephone line or not. It has two values, "on" and "off". Value "on" indicates that the user is on a phone and value "off" indicates that the user is not on any phone. If a user changes phone line

status from “on” to “off” or vice versa, the user’s PA is informed. The PA then notifies all watchers of the user.

Availability:

Parameter “availability” indicates whether a user i.e. a presentity likes to communicate with others. The value “yes” indicates that the user likes to communicate with others and “no” indicates that the user does not like to communicate with others currently. On status “yes”, other users have a higher possibility to get a successful communication to the user, while on status “no”, other users have a lower possibility to get a successful communication with the user.

Availability is a status revealing the willingness of a presentity to communicate with others. It is not the presentity’s communicating ability. On status “no”, the user does not wish to communicate with other people currently, but he can still communicate with others if necessary. For example, he can still answer emergency phone calls. If the availability status is changed from “yes” to “no”, the user’s PA is informed. The PA then notifies all watchers of this change.

Role:

Parameter “role” indicates the position status of a user. A user can have multiple role values when the user takes more than one position simultaneously. For example, Sharon works as a consultant for a company and as a professor for a university. She has two roles, “consultant” and “professor”. When the role of a user is changed from one value to another, the user’s PA is informed. The PA then notifies all watchers of the user.

If the above four parameters are all independent and the XML schema of presence extensions defines 3 locations, 3 phone line status values, 3 availability values and 6 role values, an end user can have a maximum of 3x3x3x6 = 162 possible presence status. In fact, the number of presence status may be much less than 162 because of some constrains. For an example, if “lab” is a location value, a constrain case for Sharon might

be that she cannot work in a student “lab” when she is taking the role “consultant” for her company.

4.3.2 XML Schema for the Extensions of Presence Information

The XML schema for the extensions of presence information is shown in Table. 4.1. The schema defines four parameters for presence extensions. The first parameter, “location”, has three values: “office”, “meeting room” and “car”. The second parameter, “lineStatus”, has two values: “on” and “off”. The third parameter, “role”, has five values: “doctor”, “consultant”, “professor”, “student” and “nurse”. The last parameter, “availability”, has two values: “yes” and “no”.

The schema is for a company “example.com” with namespace of “epidf”. The URI of the namespace is “urn:example-com:pidf-status-type”. The schema can be accessed through the namespace URI when a presence document needs validation.

Although the existing PIDF definition allows arbitrary elements to appear in the <status> element, it may be sometimes desirable to standardize extension status elements. The URN “urn:ietf:params:xml:ns:pidf:status” has been defined as a namespace URI for extensions standardized by the Internet Engineering Task Force (**IETF**), and new values in this namespace must be defined by a standards-track RFC.

We suppose that an RFCxxxx defines the above extensions of presence information. The schema file of the extensions should be in the namespace “urn:ietf:params:xml:ns:pidf:status”. The RFC defining the extensions should register an extension name, “pre-ex1”, within the namespace with the Internet Assigned Numbers Authority (**IANA**). The new XML schema is shown in Table. 4.2.

XML schema	Schema Explanations
`<?xml version="1.0" encoding="UTF-8"?>` `  <xs:schema targetNamespace="urn:example.com:pidf-status-type"`	namespace definition with URI
`xmlns:epidf="urn:example.com:pidf-status-type"`	schema name is “epidf”

`xmlns:xs="http://www.w3.org/2001/XMLSchema"`
`elementFormDefault="qualified"`
`attributeFormDefault="unqualified">`
	URI of the XML schema

`<xs:simpleType name="location">`
`  <xs:restriction base="xs:string">`
`  <xs:enumeration value="office"/>`
`  <xs:enumeration value="meeting room"/>`
`  </xs:restriction>`
`</xs:simpleType>`
	definition of “location”
	location is a string parameter
	value can be “office”
	value can be “meeting room”

`<xs:simpleType name="lineStatus">`
`  <xs:restriction base="xs:string">`
`  <xs:enumeration value="on"/>`
`  <xs:enumeration value="off"/>`
`  </xs:restriction>`
`</xs:simpleType>`
	definition of “lineStatus”
	lineStatus is a string parameter
	value can be “on”
	value can be “off”

`<xs:simpleType name="role">`
`  <xs:restriction base="xs:string">`
`  <xs:enumeration value="doctor"/>`
`  <xs:enumeration value="consultant"/>`
`  <xs:enumeration value="professor"/>`
`  <xs:enumeration value="student"/>`
`  <xs:enumeration value="nurse"/>`
`  </xs:restriction>`
`</xs:simpleType>`
	definition of “role”
	role is a string parameter
	value can be “doctor”
	value can be “consultant”
	value can be “professor”
	value can be “student”
	value can be “nurse”

`<xs:simpleType name="avilability">`
`  <xs:restriction base="xs:string">`
`  <xs:enumeration value="yes"/>`
`  <xs:enumeration value="no"/>`
`  </xs:restriction>`
`</xs:simpleType>`
	definition of “availability”
	availability is a string parameter
	value can be “yes”
	value can be “no”

`</xs:schema>`	

Table 4.1 XML Schema for Presence Extensions

XML schema	Schema Explanations

`<?xml version="1.0" encoding="UTF-8"?>`
`  <xs:schema`
`**targetNamespace="urn:ietf:params:xml:ns:pidf:status"**`
	namespace definition with URI

`**xmlns:pre-ex1="urn:ietf:params:xml:ns:pidf:status"**`
`xmlns:xs="http://www.w3.org/2001/XMLSchema"`
`elementFormDefault="qualified"`
`attributeFormDefault="unqualified">`
	schema name is “pre-ex1”
	URI of XML schema

`<xs:simpleType name="location">`
`<xs:restriction base="xs:string">`
`<xs:enumeration value="office"/>`
`<xs:enumeration value="meeting room"/>`
`<xs:enumeration value="car"/>`
`</xs:restriction>`
`</xs:simpleType>`
	definition of “location”
	location is a string parameter
	value can be “office”
	value can be “meeting room”
	value can be “car”

`<xs:simpleType name="lineStatus">`
`<xs:restriction base="xs:string">`
`<xs:enumeration value="on"/>`
`<xs:enumeration value="off"/>`
`</xs:restriction>`
`</xs:simpleType>`
	definition of “lineStatus”
	lineStatus is a string parameter
	value can be “on”
	value can be “off”

`<xs:simpleType name="role">`
`<xs:restriction base="xs:string">`
`<xs:enumeration value="doctor"/>`
`<xs:enumeration value="consultant"/>`
`<xs:enumeration value="professor"/>`
`<xs:enumeration value="student"/>`
`<xs:enumeration value="nurse"/>`
`</xs:restriction>`
`</xs:simpleType>`
	definition of “role”
	role is a string parameter
	value can be “doctor”
	value can be “consultant”
	value can be “professor”
	value can be “student”
	value can be “nurse”

`<xs:simpleType name="avilability">`
`<xs:restriction base="xs:string">`
`<xs:enumeration value="yes"/>`
`<xs:enumeration value="no"/>`
`</xs:restriction>`
`</xs:simpleType>`
	definition of “availability”
	availability is a string parameter
	value can be “yes”
	value can be “no”

`</xs:schema>`	

Table 4.2 XML Schema for Presence Extensions (in IETF standard format)

4.3.3 Presence Documents in PIDF

The paper “Presence Information Data Format (PIDF)” [21] is specified as a common presence data format for Common Profile for Presence (CPP)-compliant presence protocols. The PIDF defined in this paper also defines a new media type, "application/pidf+xml", to represent the XML MIME entity for PIDF, where “application” is the name of MIME media type and “pidf+xml” is the MIME subtype. The formal definition of the “application/pidf+xml” for PIDF is shown in Appendix A.

According to Ref. [21], presence information consists of presentity URL, presentity human readable comments and one or more **presence tuples**. A presence tuple consists of a **status**, an optional **communication address**, **relative priorities** of contact addresses, **timestamp** of the changed tuple and optional **human readable comments**.

Tuples, i.e. ordered sets of values, provide a way of segmenting presence information. Protocols or applications may choose to segment the presence information associated with a presentity for a number of reasons. For an example, components of the full presence information for a presentity come from distinct devices or different applications on the same device, and/or components are generated at different times. Tuples should be preferred over other manners of segmenting presence information, such as creating multiple PIDF instances.

Fig. 4.2 displays a presence document. The presentity, Stephen, is identified by the URI "sip:stephen@example.com”. The document contains the following presence information: 1) Stephen works as a doctor. 2) He is at his office. 3) He is talking on his phone. 4) He can accept instant message. 5) He likes to take other phone calls currently. The document is written in XML version, “1.0”. The structure of presence information is defined in two different namespaces. One namespace is default identified by the URI, “urn:ietf:params:xml:ns:pidf”; the other namespace for presence extensions, “epidf”, is identified by the URI, "urn:example-com:pidf-status-type”. The root element of the document is <presence>. The element <presence> contains three elements, they are <tuple>, <tuple> and <note>. The first tuple identified as “ab22f3” has five elements,

they are <status>, <contact>, <note>, <note> and <timestamp>. The status contains five elements, they are <basic> and four extension elements, i.e. <location>, <lineStatus>, <role> and <availability>. These extension elements are defined in the namespace, where "epidf" is identified by the URI "urn:example-com:pidf-status-type". The second tuple, identified as "ef10g9", has two elements only, i.e. <status> and <contact>. The status includes only one element, i.e. <basic>. Element <basic> indicates whether instant messages are acceptable. Value "open" indicates that they are acceptable and value "closed" indicates that they are not acceptable. The thesis focuses on presence and not on instant messaging.

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
    xmlns:epidf="urn:example.com:pidf-status-type"
    entity="sip:Stephen@example.com">

  <tuple id="ab22f3">
    <status>
      <basic>open</basic>
      <epidf:location> office </epidf:location>
      <epidf:linestatus> on </epidf:linestatus>
      <epidf:role> doctor </epidf:role>
      <epidf:availability> no </epidf:availability>
    </status>
    <contact priority="0.8">>Stephen@example.com</contact>
    <note xml:lang="en">Don't Disturb Please!</note>
    <note xml:lang="fr">Ne derangez pas, s'il vous plait</note>
    <timestamp>2003-11-01T16:49:29Z</timestamp>
  </tuple>

  <tuple id="ef10g9">
    <status>
      <basic>open</basic>
    </status>
    <contact priority="0.6">mailto:Stephen@example.com</contact>
  </tuple>
  <note>I'll attend conference in Ottawa next week</note>
</presence>
```

Figure 4.2 Stephen's Presence Document in PIDF

If the presence extensions are defined by a RFCxxxx and the schema of the extensions is stored in the namespace with URI of "urn:ietf:params:xml:ns:pidf:status", the above presence document in PIDF is modified as shown in Fig. 4.3.

```
<?xml version="1.0" encoding="UTF-8"?>
<presence xmlns="urn:ietf:params:xml:ns:pidf"
    xmlns:tns=" urn:ietf:params:xml:ns:pidf:status"
    entity="sip:Stephen@example.com">
  <tuple id="ab22f3">
    <status>
      <basic>open</basic>
      <tns:location> office </tns:location>
      <tns:linestatus> on </tns:linestatus>
      <tns f:role> doctor </tns:role>
      <tns f:availability> no </tns:availability>
    </status>
    <contact priority="0.8">>Stephen@example.com</contact>
    <note xml:lang="en">Don't Disturb Please!</note>
    <note xml:lang="fr">Ne derangez pas, s'il vous plait</note>
    <timestamp>2003-11-01T16:49:29Z</timestamp>
  </tuple>

  <tuple id="ef10g9">
    <status>
      <basic>open</basic>
    </status>
    <contact priority="0.6">mailto:Stephen@example.com</contact>
  </tuple>
  <note>I'll attend conference in Ottawa next week</note>

</presence>
```

Figure 4.3 Stephen's Presence Document in IETF Standard

The presence document in PIDF is included in and delivered with SIP NOTIFY messages to Stephen's watchers, which are discussed in chapter 3.

4.4 CPL Extensions for Presence

CPL extensions for presence are designed to describe presence services for end users. These extensions can also be used when describing call processing services related to presence in Internet Telephony.

In the CPL extensions for presence, we define four top-level actions to identify the types of CPL scripts, six new operations and a presence-switch to support the services whose decisions are based on the presence status of a presentity.

4.4.1 Four New Top-level Actions

Top-level actions are actions triggered by signaling events when they arrive to the SIP server. In current CPL, there are two top-level actions, "incoming" and "outgoing", for SIP INVITE messages in call handlings. Similarly, in a presence system, in order to identify the types of CPL scripts for SIP SUBSCRIBE and SIP NOTIFY messages, we define four new top-level actions in CPL extensions. They are "incoming-subscription" and "outgoing-subscription" for SIP SUBSCRIBE messages, and "incoming-notification" and "outgoing-notification" for SIP NOTIFY messages.

"Incoming-subscription" is the action performed when a SIP SUBSCRIBE message arrives and the message's destination is the script owner i.e. the presentity. The action is written in a CPL script of a presentity and is executed when the presentity receives a subscription message.

"Outgoing-subscription" is the action performed when a SIP SUBSCRIBE message arrives and the message's originator is the script owner i.e. the watcher. The action is written in a CPL script of the watcher and is executed when the watcher sends a subscription message.

"Incoming-notification" is the action performed when a SIP NOTIFY message arrives and the message's destination is the script owner i.e. the watcher. The action is written in a CPL script of the watcher and is executed when the watcher receives a notification message.

"Outgoing-notification" is the action performed when a SIP NOTIFY message arrives and the messages's originator is the script owner i.e. the presentity. The action is written

in a CPL script of the presentity and is executed when the presentity sends a notification message to a watcher.

In the processing of a presence service in CPL, the location set of CPL is initialized to empty in cases of “incoming-subscription” and “incoming-notification”; and the location set is initialized to the destination address in case of “outgoing-subscription” and “outgoing-notification”.

There are a total of six top-level actions in Extended CPL. Actions of “incoming” and “outgoing” are used in describing call processing services. Actions of “incoming-subscription”, “outgoing-subscription”, “incoming-notification” and “outgoing-notification” are used in describing presence related services. The syntax for the extended node of CPL is given in Fig. 4.4.

```
Tag:  cpl
Parameters:
   xmls    namespaces containing CPL definition files at URIs

Sub-tags:
   ancillary  See CPL Section 10 [9]
   subaction  See CPL Section 9 [9]
   outgoing   Top-level actions to take on outgoing-calls
   incoming   Top-level actions to take on incoming-calls

   incoming-subscription
            Top-level actions to take on incoming-subscription requests

   Outgoing-subscription
            Top-level actions to take on outgoing-subscription requests

   Incoming-notification
           Top-level actions to take on incoming-notification responses

   Outgoing-notification
           Top-level actions to take on outgoing-notification responses
```

Figure 4.4 Syntax of the Extended CPL Node

4.4.2 Five Operations

Three signaling operations and two non-signaling operations are defined in the thesis. The three signaling operations are “subscribe”, “notify” and “call”. The two non-signaling operations are “approve” and “accept”.

The **Subscribe** operation causes the SIP server to send a SIP SUBSCRIBE message to the specified presentity. The syntax of node “subscribe” is given in Fig. 4.5.

```
Node:    subscribe
Outputs: noanswer      Next node if subscription was not answered
                       before timeout

         redirection   Next node if subscribe attempt was redirected
         failure       Next node if call attempt failed
         default       Default next node for unspecified outputs


Parameters:  timeout      Time to try before giving up on the
                          subscription attempt

             recurse      Whether to recursively look up redirections
             ordering     What order to try the location set in.


    Output:  noanswer
Parameters:  none

    Output:  redirection
Parameters:  none

    Output:  failure
Parameters:  none

    Output:  default
Parameters:  none
```

Figure 4.5 Syntax of Node “subscribe”

The **Notify** operation causes a SIP server to send a SIP NOTIFY message to the specified watcher. The NOTIFY message contains a presence document in PIDF. The syntax of node “notify” is given in Fig. 4.6.

```
         Node:  notify
      Outputs:  noanswer     Next node if notification was not answered
                             before timeout
                redirection  Next node if notification was redirected
                failure      Next node if notification failed
                default      Default next node for unspecified outputs
   Parameters:  timeout      Time to try before giving up on the call
attempt
                recurse      Whether to recursively look up
redirections
                ordering     In what order to try the location set.

       Output:  noanswer
   Parameters:  none

       Output:  redirection
   Parameters:  none

       Output:  failure
   Parameters:  none

       Output:  default
   Parameters:  none
```

Figure 4.6 Syntax of Node “notify”

The **Call** operation includes an “accept” operation. It makes the W-PA to accept the notification and initiates a new call automatically for the watcher. The watcher i.e. the owner of the CPL script becomes a caller in this case. The operation causes the SIP server to send a SIP INVITE message to a specified callee. The operation “call” immediately terminates the execution of the CPL script, so this node has neither output nor next node. The syntax of the node “call” is given in Fig. 4.7.

```
Node:           call
Outputs:        None
Next node:      None
Parameters:     None
```

Figure 4.7 Syntax of Node “call”

The **Approve** operation tells a PA of a presentity that the watcher request is approved with a limited duration. The PA then starts to prepare the notification message. The syntax of node "approve" is shown in Fig. 4.8. The operation "approve" immediately terminates the execution of the CPL script, so the node has neither output nor next node.

```
Node:           approve
Outputs:        None
Next node:      None
Parameters:     duration
```

Figure 4.8 Syntax of Node "approve"

The **Accept** operation makes a PA of a watcher to accept the received notification. The displaying of presence information is refreshed. The "accept" operation immediately terminates the execution of the CPL script, so the node has neither output nor next node. The syntax of node "accept" is shown in Fig. 4.9.

```
Node:           accept
Outputs:        None
Next node:      None
Parameters:     None
```

Figure 4.9 Syntax of Node "accept"

4.4.3 Presence-switch

In current CPL, switches represent choices that a CPL script can make, based on an attribute of the original call request. A switch followed by conditions indicates a type of condition. Each condition corresponds to a node output. The output points to the next node to execute if the condition is true. When the script is executed, the conditions are checked in the order that they are presented in the script. The output of the first matching node is taken. The information on which a choice is based is carried in the SIP message or is easy to obtain.

"presence-switch" allows an end user to make decisions based on the presence status of a presentity. The presentity may be the user himself or somebody else. The presence

information is carried in the current SIP NOTIFY message otherwise the presence information can be obtained by some means through a presence system.

Node "presence switch" and node "presence":

A "presence-switch" allows a user to make a decision based on presence status of a presentity. The syntax of node "presence-switch" and node "presence" are shown in Fig. 4.10.

Node "presence-switch" has two mandatory parameters, "presentity" and "timeout". Parameter "presentity" identifies a presentity, and parameter "timeout" gives the CPL server a time limit to try before giving up retrieving presence information. The value of parameter "presentity" is a URI of the presentity. A CPL server can reject the CPL script while loading if its parameters cannot be resolved. If the parameter, either "presentity" or "timeout", is not present, the server automatically sets the default value for them. The default "presentity" is the current user and the default of "timeout" depends on physical servers.

Node "presence" is next to node "presence-switch". It specifies a presence status to verify whether the presentity matches the status or not. Four presence parameters, "location", "lineStatus", "role" and "availability" for presence extensions have been shown in Table. 4.1. In this case, the presence node can have a minimum of one and a maximum of four presence parameters. Node "presence" has three possible output nodes, "success", "unmatched" and "failure". Node "success" is taken if the presentity exactly matches the presence status specified in node "presence". Node "unmatched" is taken if the retrieval of the presentity's presence information succeeds but the presentity does not exactly match the specified status. Node "failure" is taken if the retrieval of the presence information fails for some reasons including the case of timeout. If an output is not present in the script, the execution of the script terminates and a default behavior is performed automatically. Clients should specify the three outputs "success", "unmatched" and "failure" in that order, so that their scripts comply with the XML DTD that will be discussed in subsection 4.4.4.

Node:	presence-switch	
Next node:	presence	specific presence status to match
Parameters:	presentity	URI of a presentity
	timeout	time to try before giving up retrieving presence info

Node:	presence	
Output	success	next node if the presence status is exactly matched
	unmatched	next node if the presence status is not exactly matched
	failure	next node if the retrieving of presence info has failed

Parameters:	location	location of a presentity “office”, value of “location” ……
	lineStaus	phone line status. “on”, value of “lineSatus” “off”, value of “lineSatus”
	role	position status. “professor”, value of “role” ……
	availability	status of willingness to communicate with others “yes”, value of “availability” “off” , value of “availability”

Output:	success
Parameters:	none
Output:	unmatched
Parameters:	none
Output:	failure
Parameters:	none

Figure 4.10 Syntax of Node “presence-switch” and Node “presence”

4.4.4 XML DTD of CPL Extensions for Presence

The definition of CPL extensions for presence is saved in the file "cplPresence.dtd" and the file can be accessed at the URI, "http://www.site.uottawa.ca/~djiang/theis/cplPresence.dtd". The definition of CPL is saved in the file "cpl.dtd" and the file can be accessed at the URI, "http://www.site.uottawa.ca/~djiang/theis/cpl.dtd". The XML definition file "cpl.dtd" for CPL is given in Appendix B. The XML definition file containing CPL extensions for presence "cplPresence.dtd" is shown in Fig. 4.11.

The switch nodes are modified to:

```
<!-- Switch nodes -->
<!ENTITY % Switch 'address-switch|string-switch|time-switch|
                   priority-switch|presence-switch' >

<!-- Switches: choices a CPL script can make. -->

   <!-- All switches can have an 'otherwise' output. -->
   <!ELEMENT otherwise ( %Node; ) >

   <!-- All switches can have a 'not-present' output. -->
   <!ELEMENT not-present ( %Node; ) >

   <!-- Presence-switch makes choices based on presence infomation. -->
   <!ELEMENT presence-switch ( presence*, (not-present, presence*)?,
                               otherwise? ) >
   <!-- <not-present> must appear at most once -->

   <!ATTLIST presence-switch
      presentity    CDATA    #REQUIRED
      timeout       CDATA    #IMPLIED
   >

   <!ELEMENT presence ( success?,unmatched?,failure? )>

   <!ATTLIST presence
      location        CDATA    #IMPLIED
      lineStatus      CDATA    #IMPLIED
      role            CDATA    #IMPLIED
      availability    CDATA    #IMPLIED

   > <!– at least one and at most four of the above attributes must
         appear -->

   <!ELEMENT success  ( %Node; ) >
   <!ELEMENT unmatched ( %Node; ) >
   <!ELEMENT failure ( %Node; ) >
```

The signaling nodes are modified to

```
   <!-- Signalling action nodes -->
      <!ENTITY % SignallingAction 'proxy|redirect|reject|
                                   call|subscribe|notify' >
```

The other action nodes are modified to

```
   <!-- Other actions -->
      <!ENTITY % OtherAction 'mail|log|approve|accept' >
```

The new added signaling nodes are defined as:

```
   <!ELEMENT subscribe (approve?,pending?,reject?,
                        noanswer?,default?)>

   <!ATTLIST subscribe
      timeout       CDATA    #IMPLIED
      recurse       (yes|no) "yes"
      ordering      CDATA    "parallel"
   >
   <!ELEMENT approve ( %Node; ) >
   <!ELEMENT pending ( %Node; ) >
   <!ELEMENT reject ( %Node; ) >
   <!ELEMENT noanswer ( %Node; ) >
   <!ELEMENT default ( %Node; ) >

   <!ELEMENT notify ( success?,noanswer?,redirection?,
                      failure?,default? ) >
   <!ATTLIST notify
      timeout       CDATA    #IMPLIED
      recurse       (yes|no) "yes"
      ordering      CDATA    "parallel"
   >
   <!ELEMENT success ( %Node; ) >
   <!ELEMENT noanswer ( %Node; ) >
   <!ELEMENT redirection ( %Node; ) >
   <!ELEMENT failure ( %Node; ) >
   <!ELEMENT default ( %Node; ) >
```

The new added other action nodes are defined as:

```
   <!ELEMENT approve EMPTY >
   <!ATTLIST approve
      duration      CDATA    #IMPLIED
   >

   <!ELEMENT accept EMPTY >
```

The new added SubActions are defined as:

```
   <!ENTITY % SubActions 'subscribe?,notify?,call?,approve?,
                          accept?' >
   <!ELEMENT subscribe ( %Node; )>
   <!ELEMENT notify ( %Node; )>
   <!ELEMENT call ( %Node; )>
   <!ELEMENT approve ( %Node; )>
```

```
      <!ELEMENT accept ( %Node; )>

   The TopLevelActions are modified to:

      <!ENTITY % TopLevelActions 'outgoing?,incoming?,
                incoming-subscription?, outgoing-subscription?,
                incoming-notification?,outgoing-notification?' >
      <!ELEMENT outgoing ( %Node;, %SubActions; )>
      <!ELEMENT incoming ( %Node;, %SubActions; )>
<!ELEMENT incoming-subscription ( %Node;, %SubActions; )>
<!ELEMENT outgoing-subscription ( %Node;, %SubActions; )>
<!ELEMENT incoming-notification ( %Node;, %SubActions; )>
<!ELEMENT outgoing-notification ( %Node;, %SubActions; )>
```

Figure 4.11 Definition of CPL Extensions for Presence

An example of how a CPL script declares the namespaces of Extended CPL is shown in Table. 4.3. In the example, Stephen approves Mary's subscription request unconditionally. Element <incoming-subscription> is defined by CPL extensions in the namespace named "cplPresence". The namespace can be accessed at "http://www.site.uottawa.ca/~djiang/thesis/cplPresence.dtd".

A CPL script	**Script Explanations**
<?xml version = "1.0" encoding = "UTF-8"> <cpl **xmlns =** "http://www.site.uottawa.ca/ ~djiang/thesis/cpl.dtd" **xmlns:cplPresence =**"http://www.site.uottawa.ca/ ~djiang/thesis/cplPresence.dtd **">** **<cplPresence:incoming-subscription>** **<**address-switch field ="origin"> <address is = "sip:mary@site.uottawa.ca"> **<cplPresence:approve/>** </address > </address-switch> </cplPresence:incoming-subscription > </cpl>	<!– actions "incoming-subscription" and action "approve" are defined in namespace "cplPresence" and the namespace can be accessed at http://www.site.uottawa.ca/~djiang/thesis/cplPresence.dtd" --!>

Table 4.3 Namespace Declarations for Extended CPL

Note that all CPL extensions that we have described are compatible with standard CPL.

4.5 End User Services Specified in Extended CPL

In a new presence system with CPL extensions, beside the presence system basic services described in chapter 3, end users can benefit of many new services for their own special needs. Assuming that the CPL definitions can be accessed at URL, "http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd" and CPL extensions for presence can be accessed at "http://www.site.uottawa.ca/~djiang/thesis/cplPresence.dtd", these new services are classified and described with examples as follow:

4.5.1 Presence Extensions

Presence information can contains not only a line indicator but also indicators of phone line status, location, role and availability status etc.

Mary is a watcher of a presentity, Stephen. Mary can know the following presence information of Stephen: Stephen is in his office working as a professor; he is talking with somebody on the phone and he does not like to answer phone calls currently. Stephen's presence status can be listed with the following parameters and values:

Location: "office"
Role: "professor"
LineStatus: "on"
Availability: "no"

4.5.2 Outgoing-subscription Services

Screening services:

An end user can have his specific screening services for outgoing-subscription requests. The screening services can be based on address, time, language, priority, string, presence status or any combinations.

Stephen sets an outgoing-subscription screening policy for security reason. All of his subscription requests will be blocked after working hours on workdays. Through the policy, nobody can use Stephen's presence service after working hours. The policy in CPL is shown in Fig. 4.12. The decision in the policy is based on time.

```
<cpl xmlns = "http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd"
    xmlns:cplPresence ="http://www.site.uottawa.ca/~djiang/thesis/
                        cplPresence.dtd ">

<cplPresence:outgoing-subscription>
  <time-switch tzid="America/New_York"
               tzurl="http://zones.example.com/tz/America/New_York">
    <time dtstart="20000703T170000" duration="PT14H"
               freq="weekly" byday="MO,TU,WE,TH,FR">
      <reject/>
    </time>
  </time-switch>
 </cplPresence:outgoing-subscription>
</cpl>
```

Figure 4.12 Screening Outgoing-subscriptions based on Time

```
<cpl xmlns = "http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd"
    xmlns:cplPresence ="http://www.site.uottawa.ca/~djiang/thesis/
                        cplPresence.dtd ">

<cplPresence:outgoing-subscription>
  <address-switch field ="original-destination">
    <address is = "sip:presenceServices@example.com">

      <cplPresence:presence-switch presentity =
                                    "sip:Sharon@example.com">
        <cplPresence:presence role = "consultant">
          <cplPresence:success>
            <cplPresence:subscribe/>
          </cplPresence:success>

          <otherwise>
            <reject/>
          </otherwise>
        </cplPresence:presence>
      </cplPresence:presence-switch >

    </address >
  </address-switch>
</cplPresence:outgoing-subscription>
</cpl>
```

Figure 4.13 Screening Outgoing-subscriptions based on Address & Presence Status

Sharon works as a "consultant" for a company and as a "professor" for a university. Sharon can benefit of the presence services of the company only when she is taking the role of "consultant", mostly in afternoons. Sharon writes a self-imposed policy i.e. an outgoing-subscription screening policy. The policy is shown in Fig. 4.13. In the policy, she can send a successful subscription request only when she takes the role of "consultant" for the company; otherwise the request is screened out. The policy is based on combined decisions of address and presence status.

Forwarding services:

Stephen knows that Sharon works as a consultant for a company between 9:00am and 12:00am on workdays. Stephen's requests will be automatically forwarded to Sharon's URI of her company between 9:00am and 12:00am on workdays. Stephen's policy is shown in Fig. 4.14.

```
<cpl xmlns = "http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd"
    xmlns:cplPresence ="http://www.site.uottawa.ca/~djiang/thesis/
                        cplPresence.dtd ">

<cplPresence:outgoing-subscription>
  <address-switch>
    <address is="sip:Sharon@site.uottawa.ca">

      <time-switch tzid="America/New_York"
           tzurl="http://zones.example.com/tz/America/New_York">
        <time dtstart="20000703T090000" duration="PT3H"
              freq="weekly" byday="MO,TU,WE,TH,FR">
          <location url="sip:Sharon@example.com">
            <proxy/>
          </location>

        </time>
      </time-switch>

    </address>
  </address-switch>
</cplPresence:outgoing-subscription>

</cpl>
```

Figure 4.14 Forwarding Outgoing-subscriptions based on Address & Time

4.5.3 Incoming-subscription Services

Screening services:

An end user can have screening services for incoming-subscription requests. These screening services can be based on address, time, language, priority, string, a presence status or any combinations.

Sharon rejects subscription requests from Mary unconditionally. The incoming-subscription screening service is based on address. Sharon's policy is shown in Fig. 4.15.

```
<cpl xmlns = "http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd"
    xmlns:cplPresence ="http://www.site.uottawa.ca/~djiang/thesis/
                        cplPresence.dtd ">

<cplPresence:incoming-subscription>
  <address-switch field="origin">
    <address is="sip:mary@example.com">
      <reject/>
    </address >
  </address-switch>
  </cplPresence:incoming-subscription>
</cpl>
```

Figure 4.15 Screening Incoming-subscriptions based on Address

```
<cpl xmlns = "http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd"
    xmlns:cplPresence ="http://www.site.uottawa.ca/~djiang/thesis/
                        cplPresence.dtd ">

<cplPresence:incoming-subscription>
  <time-switch tzid="America/New_York"
               tzurl="http://zones.example.com/tz/America/New_York">
    <time dtstart="20000703T170000" duration="PT16H"
               freq="weekly" byday="MO,TU,WE,TH,FR">
      <reject/>
    </time>
  </time-switch >
</cplPresence:incoming-subscription>

</cpl>
```

Figure 4.16 Screening Incoming-subscriptions based on Time

The incoming-subscription service based on time is shown in Fig. 4.16. Sharon rejects subscription requests after working hours.

Screening based on the combination of address and presence status is shown in Fig. 4.17. Sharon rejects subscription requests from Peter if she is working as a professor for a university.

```
<cpl xmlns = "http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd"
    xmlns:cplPresence ="http://www.site.uottawa.ca/~djiang/thesis/
                        cplPresence.dtd ">

<cplPresence:incoming-subscription>
  <address-switch field="origin">
    <address is="sip:peter@example.com">
      <cplPresence:presence-switch presentity =
                                    "sip:Sharon@example.com">
        <cplPresence:presence role = "professor">
          <cplPresence:success>
            <reject/>
          </cplPresence:success>
        </cplPresence:presence>
      </cplPresence:presence-switch>
    </address>
  </address-switch>
</cplPresence:incoming-subscription>
</cpl>
```

Figure 4.17 Screening Incoming-subscriptions based on Address & Presence status

Conditionally approving services:

The approving services of a presentity can be based on address, time, language, priority, presence status or any combinations.

In the example shown in Fig. 4.18, Sharon only accepts her boss's subscription requests from 9:00am to 5:00pm, Monday to Friday.

```
<cpl xmlns = "http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd"
    xmlns:cplPresence ="http://www.site.uottawa.ca/~djiang/thesis/
                        cplPresence.dtd ">

<cplPresence:incoming-subscription>
  <address-switch field ="origin">
    <address is = "sip:SharonBoss@example.com">
      <time-switch tzid="America/New_York"
                   tzurl="http://zones.example.com/tz/America/New_York">

        <time dtstart="20000703T090000" duration="PT8H"
              freq="weekly" byday="MO,TU,WE,TH,FR">
          <cplPresence:approve/>
        </time>

        <otherwise>
          <reject/>
        </otherwise>

      </time-switch >
    </address>
  </address-switch>
</cplPresence:incoming-subscription>
</cpl>
```

Figure 4.18 Conditionally Approving Incoming-subscriptions

```
<cpl xmlns = "http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd"
    xmlns:cplPresence ="http://www.site.uottawa.ca/~djiang/thesis/
                        cplPresence.dtd ">

<cplPresence:incoming-subscription>
  <address-switch field="origin">
    <address is="sip:peter@example.com">
      <cplPresence:presence-switch presentity =
                                    "sip:Sharon@example.com">
        <cplPresence:presence role = "professor">
          <cplPresence:success>
            <location url="sip:Sharon@site.uottawa.ca">
              <proxy />
            </location>
          </cplPresence:success>
        </cplPresence:presence>
      </cplPresence:presence-switch>
    </address>
  </address-switch>

</cplPresence:incoming-subscription>
</cpl>
```

Figure 4.19 Forwarding Incoming-subscriptions

Forwarding services:

A presentity can forward a subscription request to another URL. The URL can be the URL of himself or somebody else. The forwarding services can be based on address, time, language, message priority, presence status or any combinations.

The example for the forwarding service is shown in Fig. 4.19. In the policy, Peter's requests to Sharon will be automatically forwarded to her URI at the university if she is working as a professor for the university.

4.5.4 Outgoing-notification Services

Screening services:

A presentity can refuse to notify a watcher if he wants to hide his currently updated presence status. The service for privacy can be based on address, time, presence status or any combinations.

In the policy shown in Fig. 4.20, Peter is a watcher of Sharon. All of Sharon's watchers except Peter will be notified when Sharon makes a phone call. Sharon blocks Peter from knowing that she is on the phone.

```
<cpl xmlns = "http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd"
    xmlns:cplPresence ="http://www.site.uottawa.ca/~djiang/thesis/
                        cplPresence.dtd ">
<cplPresence:outgoing-notification>
  <address-switch field="origin-destination">
    <address is="sip:peter@example.com">
      <cplPresence:presence-switch presentity =
                                    "sip:Sharon@example.com">
        <cplPresence:presence lineStatus = "on">
          <cplPresence:success>
            <reject/>
          </cplPresence:success>
        </cplPresence:presence>
      </cplPresence:presence-switch>
    </address>
  </address-switch>
</cplPresence:outgoing-notification>
</cpl>
```

Figure 4.20 Screening Outgoing-notifications

Notification frequency services:

When presence information is updated too frequently, a presentity can control the frequency of notifications. This service can be based on address, time, language, priority, presence status of a presentity or any combinations.

In the following example shown in Fig. 4.21, the notification frequency from Sharon to Stephen is no more than once every five minutes if the presence information of Sharon is updated more frequently than every five minutes.

```
<cpl xmlns = "http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd"
    xmlns:cplPresence ="http://www.site.uottawa.ca/~djiang/thesis/
                        cplPresence.dtd ">

<cplPresence:outgoing-notification>

  <address-switch field="origin-destination">
    <address is="sip:peter@example.com">

      <time-switch tzid="America/New_York"
           tzurl="http://zones.example.com/tz/America/New_York">
        <time dtstart="20000703T090000" duration="PT8H"
              freq="weekly" byday="MO,TU,WE,TH,FR" byminute="5">
          <cplPresence:notify/>
        </time>
      </time-switch>

    </address>
  </address-switch>

</cplPresence:outgoing-notification>

</cpl>
```

Figure 4.21 Controlling Outgoing-notification Rate

If we combine the situation shown in Fig. 4.23 with the situation shown in Fig. 4.24, a presentity can notify his watchers differently. When Sharon's presence information is updated, she can refuse to notify Peter when she is on her phone. She can notify Stephen with a frequency of less than every five minutes. She can notify her mum unconditionally etc.

4.5.5 Incoming-notification Services

Screening service:

A watcher can have incoming-notification screening services. The services can be based on address, time, language, priority, presence status or any combinations.

Peter is unlikely to accept too frequent notifications from Sharon when Sharon's presence status changes too frequently. Peter can accept a maximum of one notification every 5 minutes. The example is shown in Fig. 4.22.

```
<cpl xmlns = "http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd"
    xmlns:cplPresence ="http://www.site.uottawa.ca/~djiang/thesis/
                        cplPresence.dtd ">

<cplPresence:incoming-notification>
  <address-switch field="origin">
    <address is="sip:Sharon@example.com">
      <time-switch tzid="America/New_York"
                   tzurl="http://zones.example.com/tz/America/New_York">
        <time dtstart="20000703T090000" duration="PT8H"
              freq="weekly" byday="MO,TU,WE,TH,FR" byminute="5">

          <cplPresence: accept/>
        </time>
      </time-switch>

      <otherwise>
        <reject/>
      </otherwise>

    </address>
  </address-switch>
</cplPresence:incoming-notification>
</cpl>
```

Figure 4.22 Screening Incoming-notifications

Forwarding services:

A watcher can forward a notification to another location. The location can be a location of the watcher himself or of somebody else. The forwarding services can be based on the address, time, language, priority, presence status or any combinations.

Peter forwards all of his incoming-notifications to the computer in his office when he is in the office. The example is shown in Fig. 4.23.

```
<cpl xmlns = "http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd"
    xmlns:cplPresence ="http://www.site.uottawa.ca/~djiang/thesis/
                        cplPresence.dtd ">

<cplPresence:incoming-notification>
  <cplPresence:presence-switch presentity = "sip:peter@example.com">
    <cplPresence:presence location = "office">
      <cplPresence:success>
        <location url="sip:peter@innet.example.com">
          <proxy/>
        </location>
      </cplPresence:success>
    </cplPresence:presence>
  </cplPresence:presence-switch>
</cplPresence:incoming-notification>
</cpl>
```

Figure 4.23 Forwarding Incoming-notifications based on Presence Status

Automatic phone call:

A watcher can make an automatic phone call on the basis of address, time, language, priority, presence status or any combinations.

Peter would make an automatic call to Sharon as soon as Peter receives Sharon's notification and knows that Sharon is in her office. The policy is shown in Fig. 4.24.

All of the above examples are end user services described in Extended CPL. This gives an idea of what the end user services might be in a presence system. These services are classified into four types. The four types are outgoing-subscription, incoming-subscription, outgoing-notification and incoming-notification. These services can be based on time, address, priority, language, string, presence status or any combinations. With CPL extensions for presence, end users can benefit of many services with much flexibility.

```
<cpl xmlns = ”http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd”
    xmlns:cplPresence ="http://www.site.uottawa.ca/~djiang/thesis/
                        cplPresence.dtd ">

<cplPresence:incoming-notification>
  <address-switch field="origin">
    <address is="sip:Sharon@example.com">

      <cplPresence:presence-switch presentity =
“sip:Sharon@example.com”>
        <cplPresence:presence location = "office">
          <cplPresence:success>

            <location url="sip:Sharon@example.com">
              <cplPresence:call/>
            </location>

          </cplPresence:success>
        </cplPresence:presence>
      </cplPresence:presence-switch>

    </address>
  </address-switch>
</cplPresence:incoming-notification>

</cpl>
```

Figure 4.24 Automatic Calls based on Address and Presence Status

4.6 Call Processing Services Related to Presence

Presence services can be widely used in a call processing system. These services will much enrich end user features in Internet Telephony. With CPL extensions for presence, end users can get not only the current features described in [9] but also many new services related to presence. In this section, some new services are classified and illustrated through various examples.

4.6.1 Outgoing-call Services

Screening services:

Outgoing-call screening services can be based on address, time, language, priority, string, presence status or any combinations.

Sharon does not like her outgoing calls forwarded to voice mail. If her boss's availability status is "no", it is most likely that her call to the boss is forwarded to voice mail. Sharon blocks her calls to her boss when the boss is talking on his phone and his availability status is "no". The example is shown in Fig. 4.25.

```
<cpl xmlns = "http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd"
    xmlns:cplPresence ="http://www.site.uottawa.ca/~djiang/thesis/
                        cplPresence.dtd ">

<outgoing>
  <address-switch field ="original-destination">
  <address is = "sip:SharonBoss@example.com">

    <cplPresence:presence-switch presentity=
                                    "sip:SharonBoss@example.com">
    <cplPresence:presence lineStatus = "on" availability = "no">
      <cplPresence:success>
        <reject/>
      </cplPresence:success>
    </cplPresence:presence>
    </cplPresence:presence-switch >

  </address >
  </address-switch>

</outgoing>
</cpl>
```

Figure 4.25 Screening Outgoing-calls based on callee's Presence Status

Sharon is a consultant for a company and a professor for a university. She can benefit of the telephone services of the company for free only when she is taking the role of "consultant".

She needs a self-imposed outgoing-call screening services. She can successfully use the telephone services only when she is working as a consultant for the company, otherwise her calls for the services will be blocked automatically. The policy is shown in Fig. 4.26.

```
<cpl xmlns = "http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd"
    xmlns:cplPresence ="http://www.site.uottawa.ca/~djiang/thesis/
                        cplPresence.dtd ">
<outgoing>
  <address-switch field ="original-destination">
    <address is = "sip:companyServices@example.com">
      <cplPresence:presence-switch presentity =
                                      "sip:Sharon@example.com">
        <cplPresence:presence role = "consultant">
          <cplPresence:success>
            <proxy/>
          </cplPresence:success>
        </cplPresence:presence>
        <otherwise>
          <reject/>
        </otherwise>
      </cplPresence:presence-switch >
    </address >
  </address-switch>
</outgoing>
</cpl>
```

Figure 4.26 Screening Outgoing-calls based on Caller's Presence Status

```
<cpl xmlns = "http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd"
     xmlns:presence = "http://www.site.uottawa.ca/~djiang/thesis/
                       cplPresence.dtd">

<outgoing>
  <address-switch field ="original-destination">
    <address is = "sip:secretary@example.com">
      <cplPresence:presence-switch presentity = "sip:CEO@example.com">
        <cplPresence:presence location = "office">
          <cplPresence:success>
            <proxy/>
          </cplPresence:success>
        </cplPresence:presence>
        <otherwise>
          <reject/>
        </otherwise>
      </cplPresence:presence-switch >
    </address >
  </address-switch>
</outgoing>
</cpl>
```

Figure 4.27 Screening Outgoing-calls based on third party's Presence Status

Sharon needs her CEO's signature for her promotion application form. A secretary keeps Sharon's official form. Sharon sets an outgoing-call screening service. If the CEO's is in his office, she can call the secretary successfully to remind her to get CEO's signature, otherwise her call is blocked. Sharon's policy is shown in Fig. 4.27. Through the policy, Sharon can get the CEO's signature as soon as possible without disturbing the secretary unnecessarily.

Forwarding services:

Outgoing-call forwarding services can be based on address, time, language, priority, string, presence status or their combinations.

In the example shown in Fig. 4.28, Sharon works for a university when she is taking the role of "professor", and she works for a company when she is taking the role of "consultant". Stephen's calls to Sharon are automatically forwarded to her URI at the company when she is working for her company.

```
<cpl xmlns = ”http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd”
     xmlns:presence = “http://www.site.uottawa.ca/~djiang/thesis/
                       cplPresence.dtd”>

<outgoing>
  <address-switch field =“original-destination”>
    <address is = “sip:Sharon@site.uottawa.ca”>

      <cplPresence:presence-switch presentity =
                                    “pre:Sharon@example.com”>
        <cplPresence:presence role = "consultant">
          <cplPresence:success>
            <location url="sip:Sharon@example.com">
              <proxy/>
            </location>
          </cplPresence:success>
        </cplPresence:presence>
      </cplPresence:presence-switch >

    </address >
  </address-switch>
</outgoing>

</cpl>
```

Figure 4.28 Forwarding Outgoing-calls based on Address and Presence Status

4.6.2 Incoming-call Services

Incoming-call services can be based on address, time, language, priority, string, presence status or any combinations. The presentity can be a caller, a callee or a third party.

Screening services:

Sharon works for her company, "example.com", only when she takes the role of "consultant" and works for her university only when she takes the role of "professor". Sharon writes the incoming-call screening policy shown in Fig. 4.29. In the policy, all calls from her company will be blocked unless she takes the role of "consultant".

```
<cpl xmlns = "http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd"
     xmlns:presence = "http://www.site.uottawa.ca/~djiang/thesis/
                       cplPresence.dtd">

<incoming>
  <address-switch field ="origin">
    <address contains = "example.com">

      <cplPresence:presence-switch presentity =
                                    "sip:Sharon@example.com">
        <cplPresence:presence role = "consultant">
          <cplPresence:success>
            <proxy/>
          </cplPresence:success>
        </cplPresence:presence>

        <otherwise>
          <reject/>
        </otherwise>
      </cplPresence:presence-switch >

    </address >
  </address-switch>

</outgoing>
</cpl>
```

Figure 4.29 Screening Incoming-calls based Callee's Presence Status

Sharon is very cautious. She writes a safe policy for her boyfriend, Mike. She will not take Mike's call when he is driving in his car in working hours. Sharon's policy is shown in Fig.4.30.

```
<cpl xmlns = ”http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd”
     xmlns:presence = “http://www.site.uottawa.ca/~djiang/thesis/
                       cplPresence.dtd”>

<incoming>
  <address-switch field =“origin”>
    <address is = “sip:Mike@example.com”>

      <time-switch tzid="America/New_York"
           tzurl="http://zones.example.com/tz/America/New_York">
        <time dtstart="20000703T090000" duration="PT8H"
              freq="weekly" byday="MO,TU,WE,TH,FR">

          <cplPresence:presence-switch presentity =
                                       “sip:Mike@example.com”>
            <cplPresence:presence location = "car">
              <cplPresence:success>
                <reject/>
              </cplPresence:success>
            </cplPresence:presence>
          </cplPresence:presence-switch >
        </time>
      </time-switch>

    </address >
  </address-switch>

</outgoing>
</cpl>
```

Figure 4.30 Screening Incoming-calls based on Caller’s Presence Status

Forwarding services:

Sharon likes to forward all her incoming-calls to her voice mail when she is talking on her phone in her office with her availability status “no”. She likes to deal with these voice mails at a later time. Her policy is shown in Fig. 4.31.

```
<cpl xmlns = ”http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd”
     xmlns:presence = “http://www.site.uottawa.ca/~djiang/thesis/
                       cplPresence.dtd”>

<incoming>
  <cplPresence:presence-switch presentity = “sip:Sharon@example.com”>
    <cplPresence:presence location = “office” lineStatus = “on”
                          availability = “no”>
      <cplPresence:success>
        <location url=“sip:SharonVoiceMail@example.com”>
          <proxy/>
        <location>
      </cplPresence:success>
    </cplPresence:presence>
  </cplPresence:presence-switch >
</incoming>
</cpl>
```

Figure 4.31 Forwarding Incoming-calls based on Callee’s Presence Status

Dan works for a call center and he is ranked as number one on customer services. Tom is ranked as number 2 and Mary is ranked as number 3. According to the center’s policy, one employer can only serve one call at a time and Dan has the highest priority to take a customer’s phone call, followed by Tom and then Mary and so on. The centers set the incoming-call policies. Dan’s policy is shown in Fig. 4.32 and Tom’s policy is shown in Fig 4.33.

```
<cpl xmlns = "http://www.ietf.org/rfc/rfcxxxx.txt"
     xmlns:presence = “http://www.site.uottawa.ca/~djiang/thesis/
                       cplPresence.dtd”>

<incoming>
  <cplPresence:presence-switch presentity = “sip:dan1@example.com”>
    <cplPresence:presence lineStatus = "on">
      <cplPresence:success>
        <location url=“sip:tom2@example.com”>
          <proxy/>
        </location>
      </cplPresence:success>
    </presence:presence>
  </presence:presence-switch >
</incoming>
</cpl>
```

Figure 4.32 Forwarding Incoming-calls based on Callee’s Presence Status

```
<cpl xmlns = ”http://www.site.uottawa.ca/~djiang/thesis/cpl.dtd”
     xmlns:presence = “http://www.site.uottawa.ca/~djiang/thesis/
                       cplPresence.dtd”>


<incoming>
  <cplPresence:presence-switch presentity = “sip:tom2@example.com”>
    <cplPresence:presence lineStatus = "on">
      <cplPresence:success>

        <location url=“sip:mary3@example.com”>
          <proxy/>
        <location>

      </cplPresence:success>
    </cplPresence:presence>
  </cplPresence:presence-switch >
</incoming>
</cpl>
```

Figure 4.33 Forwarding Incoming-calls based on Callee’s Presence Status

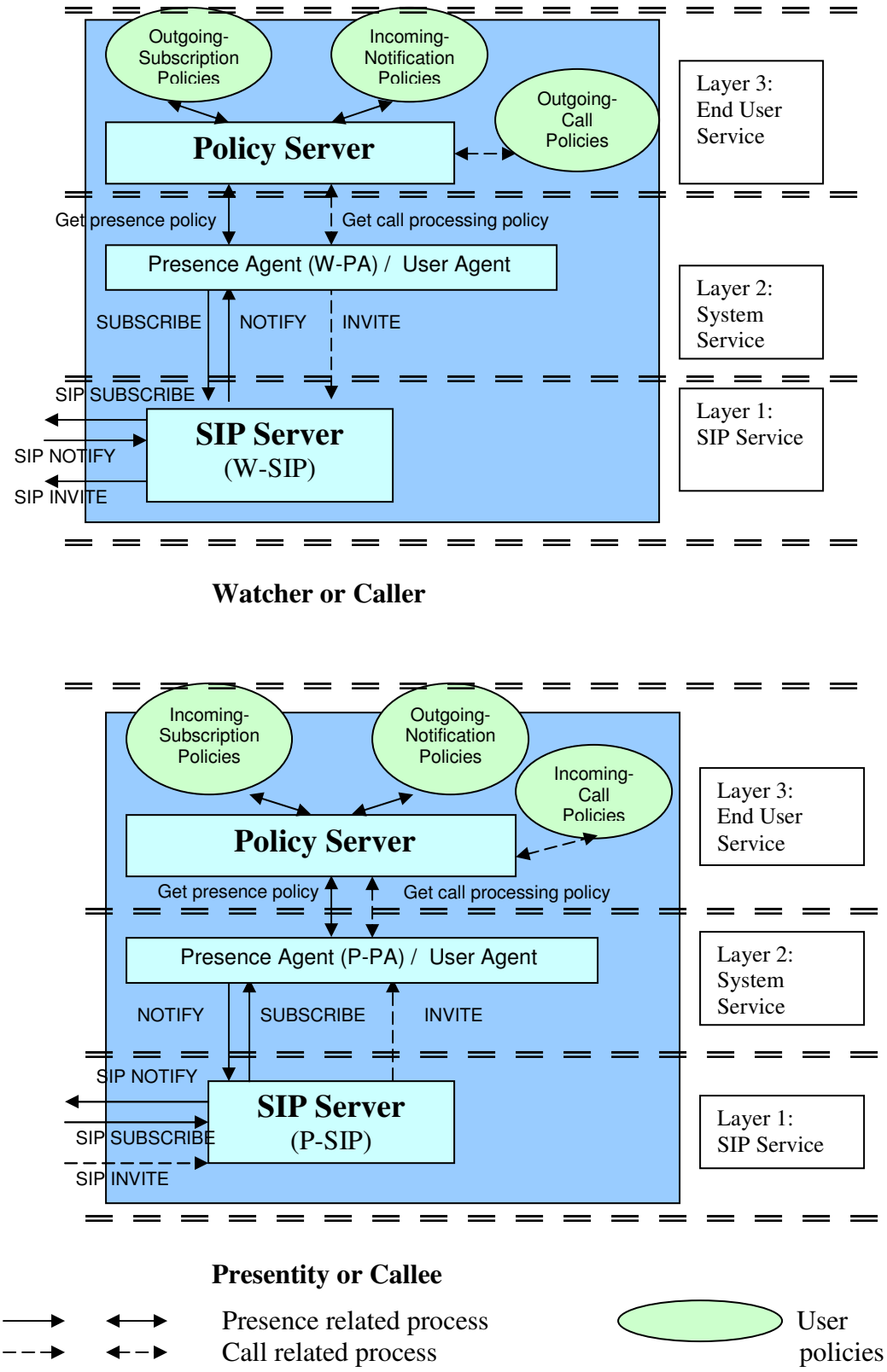

Figure 4.34 End User Policies in a three-layer Architecture

We have seen that extended CPL can be used in describing end user services in presence systems and call processing systems. These policies are written in Extended CPL and are put in policy servers. These servers are responsible of validating CPL scripts when they are loaded. PAs and UAs of end users can access these policies through the policy server at running time. The position of these policies is shown in Fig. 4.34. As a caller, the user can have outgoing-call policies; as a callee, the user can have incoming-call policies; as a watcher, the user can have outgoing-subscription policies and incoming-notification policies; as a presentity, the user can have incoming-subscription policies and outgoing-notification policies.

4.7 Conclusion

CPL and presence information extensions are discussed in this chapter, which includes explanations on why end user services in a presence system are described in extended CPL, and how to extend presence information and CPL for presence. The new system with CPL extensions for presence is presented through various application examples, and presence related services in a call processing system are also illustrated. Chapter 5 presents the implementation of a simulation system for Internet Telephony based on these principles.

Chapter 5 Simulation System

5.1 Introduction

Basic services in presence systems were described in Chapter 3. End user's specific presence services and call processing services related to presence were discussed in Chapter 4. In this chapter, a simulation system of these Internet Telephony services will be presented.

The system design is introduced section 5.2. Presence system, policy system and call processing system are described in section 5.3, section 5.4 and section 5.5 respectively. End user specific presence services and call processing services are illustrated in section 5.6. Section 5.7 is the conclusion of this chapter.

5.2 Simulation System Design

This simulation system provides a demonstration of Internet Telephony services specified in Extended CPL. These services include presence services and call processing services that might be related to presence. End users can specify policies that describe these services through Graphic User Interfaces (GUIs). These policies are translated into Extended CPL and saved in files automatically by the system. End users can access identical services through the Internet at any location.

The system is implemented in the Java programming language because of its suitability for web applications. This simulation is a web-based application that makes it accessible anywhere through the Internet.

Note that, since our system is a simulation system, rather than an implementation, no perfect mapping will be established between the components of Fig. 4.34 and our system. To demonstrate the theory described chapter 3, 4 and 5, we have only provided some functionalities of these components in our simulation system.

5.2.1 System Introduction

The simulation contains three subsystems: the presence system, the call processing system and the policy system (see Fig. 5.1). Each system name, followed by its functionalities, is a link to the system GUI. The three systems can either be independent of each other or cooperate with each other.

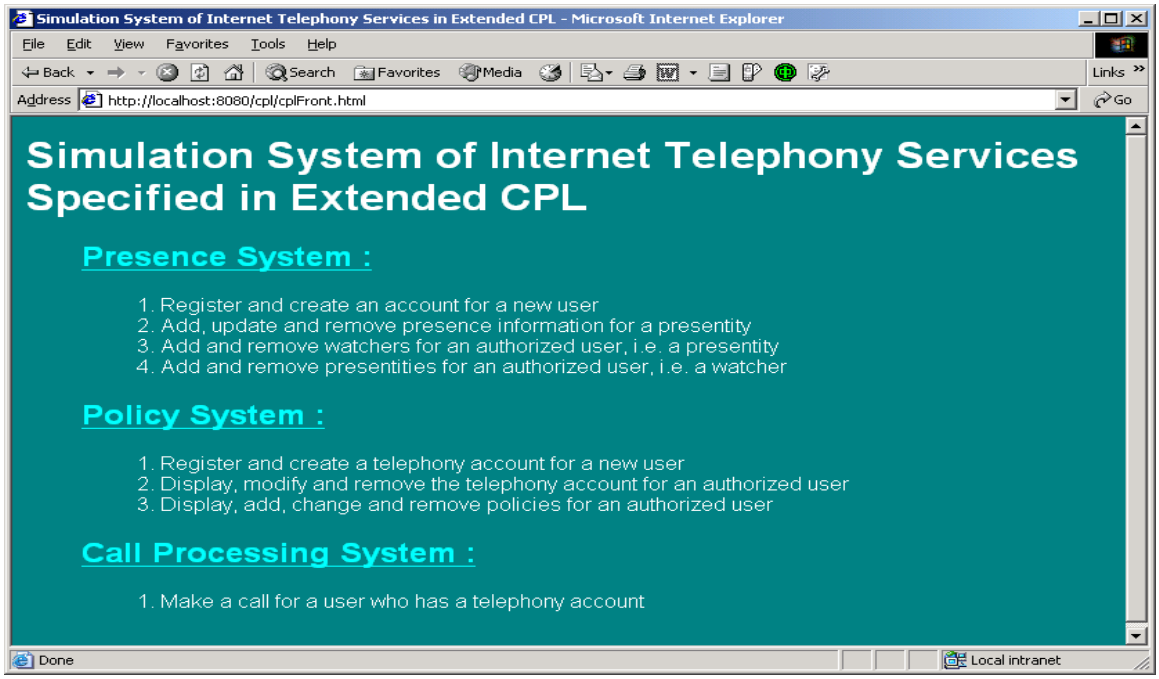

Figure 5.1 Simulation System

The presence system provides presence services, the call processing system provides call-handling services and the policy system allows end users to manage their specific services, both for presence and for call handling. In either the presence system or the call processing system, services are provided according to the users' policies. If users don't have any policies, only basic system services (i.e. default services) will be provided.

5.2.2 System Structure

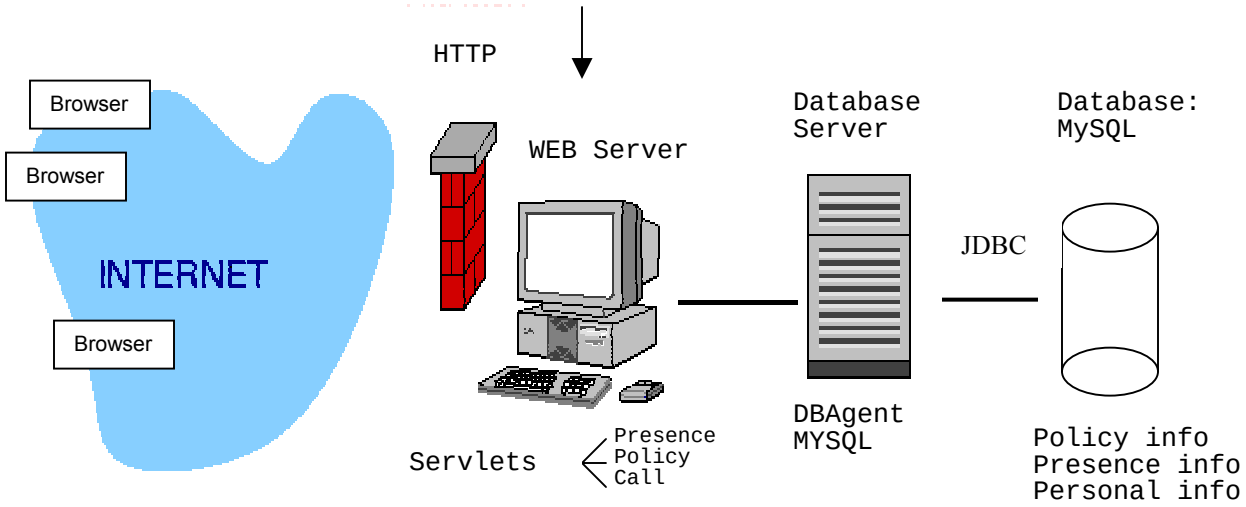

Figure 5.2 Architecture of Simulation System

The architecture of the simulation system is shown in Fig. 5.2. The architecture contains four parts: the Internet browsers, the web server, the database server and the database. The Internet browsers let users enter their requests and display request results. The web server holds Java servlets that are the central controllers that manage the system services. The Database server holds DBAgent. DBAgent works as an agent for the Database "MySQL" and has many methods to operate on the database. The database contains all service-required information. When an end user requests a service, request information is entered in the GUIs via the Internet browsers. The Java servlets accept the request information through Hyper Text Transfer Protocol (**HTTP**) and deliver it to the database agent. The database agent connects to the database "MySQL" via the Java Database Connection (**JDBC**). It gets the required data from the database and sends it back to the Java servlets. The Java servlets process the data and display the readable service results at the user's browser via HTTP.

The advantage of choosing this architecture is that the database agent and the Java servlets can reside on different computers. The database server and the web servers can

be at different locations. End users can get the identical services anywhere through the Internet.

5.2.3 Database Design

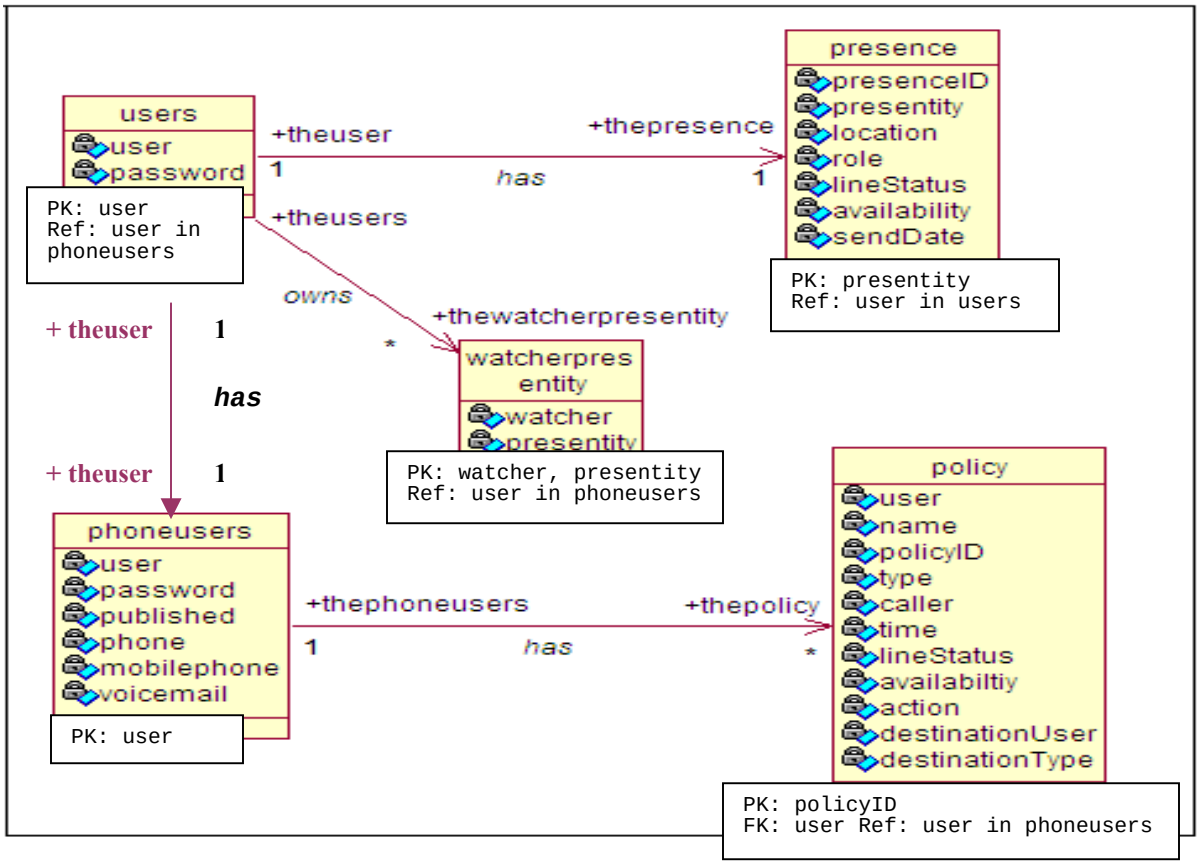

Figure 5.3 Database Schema Relationship Diagram

The database shown in Fig. 5.3 has five tables. Table “users” stores registered users’ information for the presence system. Table “presence” stores presence information of presentities. Table “watcherpresentity” stores name pairs of watchers and presentities. Table “phoneusers” stores the personal information of registered telephone users. Table “policy” stores the policy information of registered phone users. The relations among the five tables are:

1. In table “presence”, the primary key “presentity” refers to “user”, the primary key in table “users”. One user in the table “users” can have one presentity in table “presence”.

2. In table “watcherpresentity”, the primary key, “watcher” and “presentity”, refers to “user”, the primary key in table “users”. One user in table “user” can have either multiple watchers or multiple presentities in table “watcherpresentity”.

3. In table “users”, the primary key “user” references “user”, the primary key in table “phoneusers” and vice-versa. One user in table “users” corresponds to the same user in table “phoneusers” and vice-versa.

4. In table “policy”, the primary key is “policyID”. The foreign key “user” references “user”, the primary key in table “phoneusers”. One user in table “phoneusers” can have multiple policyIDs in table “policy”. Each policyID stands for one policy.

Based on the database design, one user can have only one unique name registered in the simulation system. The user can be a watcher, a presentity or can be a presentity and a watcher at the same time. If the user is a presentity, the presence information displays his current presence status. The user can have one phone account; however, he/she can have multiple policies.

5.2.4 Java File Organization

Each Java file is a Java class under a package. The simulation software structure is shown in Fig. 5.4. There are three packages in the software: a root package “cpl” and two sub-packages “cpl.app” and “cpl.web”. Root package “cpl” contains the class “DBAgent” and the two sub-packages. The DBAgent acts as the front end for the database. It has generic methods to perform several database operations. Package “cpl.web” contains the Java servlets that work as controllers. The Java servlets process service requests and return

service results back to users' browsers via HTTP. Package "cpl.app" contains the Java classes that are responsible for policy processing tasks. Each name, either a package name or a class name, is a link to the detail description. The Java Application Programmer's Interfaces (Java APIs) are included in the documents that are available with the simulation software.

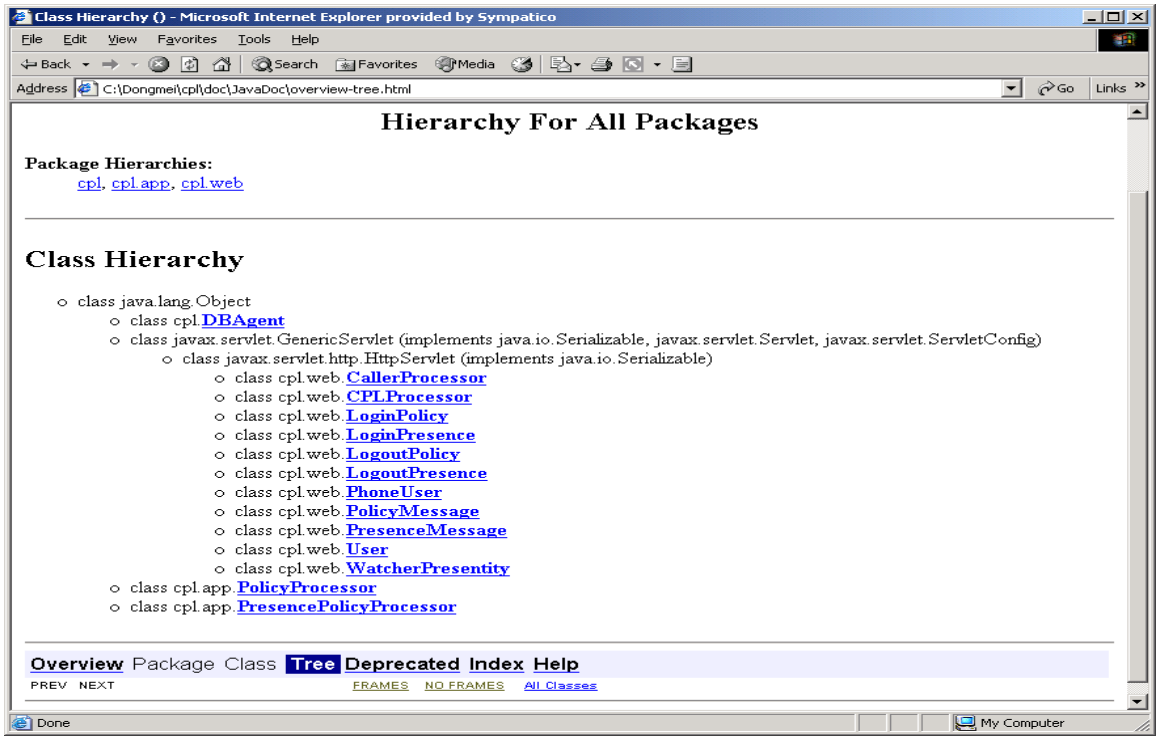

Figure 5.4 Hierarchy for all Packages

5.3 Presence System

5.3.1 Introduction

The presence system offers presence services to users. A user can be a watcher, a presentity or both at the same time. As a watcher, a user can manage his presentities; as a presentity, the user can manage his watchers and notify them of his presence information.

The root GUI of presence system is shown in Fig. 5.5. A new user has to register to the system to get an account before he can login to the system. The new user needs to fill in his name and password at registration and the personal information that is required for login to the system. After the user has registered to the system, the system sets an account for the user in the system database.

Through the GUI "presence system", a registered user can send a subscription request to a presentity. The required information is the user's name, password and the presentity's name. The user name and password are used for authorization and the presentity's name is used to identify the destination of the request.

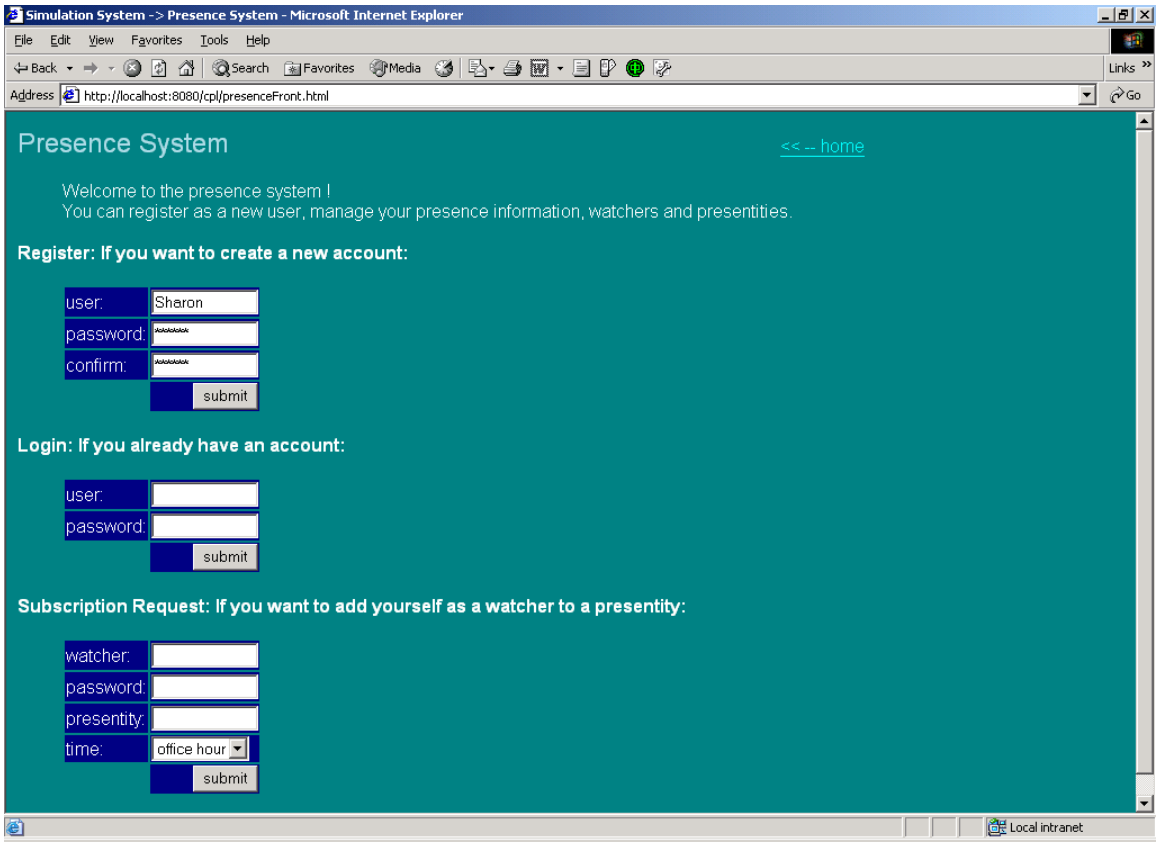

Figure 5.5 Presence System

5.3.2 The Relationship between the GUI and Presence Services

On the GUI for presence management (see Fig. 5.6), each button connects to a service of the presence system. For example, the button "add watcher" allows the current user to approve an incoming subscription request from the specified watcher. The button "adding presentity" allows the user to send out a subscription request to the specified presentity. The relationship between the GUI buttons and their corresponding services is listed in Table 5.1.

GUI Button	Presence Service
Presence information	
create presence	Register presence information to the system
edit	Update presence information in the system
delete	Remove presence information from the system
watchers	
add watcher	Process an incoming subscription request from the watcher
delete watcher	Terminate the watcher's presence service
presentities	
add presentity	Send out a subscription request to the presentity
delete	Un-subscribe to the presentity's presence service

Table 5.1 Relationship between the GUI and Presence Services

5.3.3 Presence Services

The simulation system offers the following presence services:

1) register and create an account for a new user;
2) register, modify and remove presence information for a presentity;
3) add and remove watchers for a presentity (multiple watchers are allowed);
4) add and remove presentities for a watcher (multiple presentities are allowed).

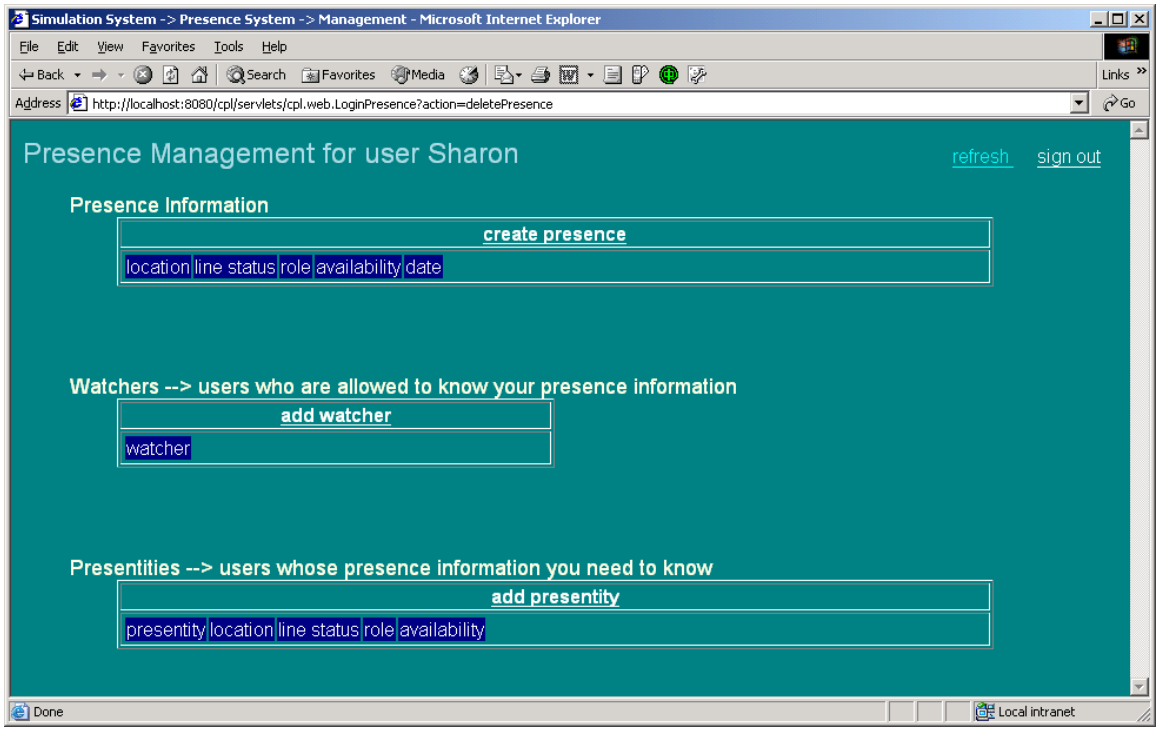

Figure 5.6 Presence Management

After a user has logged into the system, the presence system displays the GUI "Presence Management", as shown in Fig. 5.6. The presence system provides several services to end users via the GUI. For example, user Sharon can request some services as described in the following scenarios:

Scenario 1: Sharon registers her presence information

Sharon clicks the button "create presence" and the system displays the GUI "New Presence" (see Fig. 5.7). Through the GUI, Sharon fills the following presence information in office hour (i.e. 8:00am to 4:00pm):

1) Location: "office" (She is in her office);
2) Line Status: "on" (She is talking on her phone);
3) Role: "doctor" (She works as a doctor currently);
4) Availability Status: "no" (She does not want to communicate with others currently).

After her presence information is submitted, as a presentity, Sharon has successfully registered her presence information to the system (see Fig. 5.8).

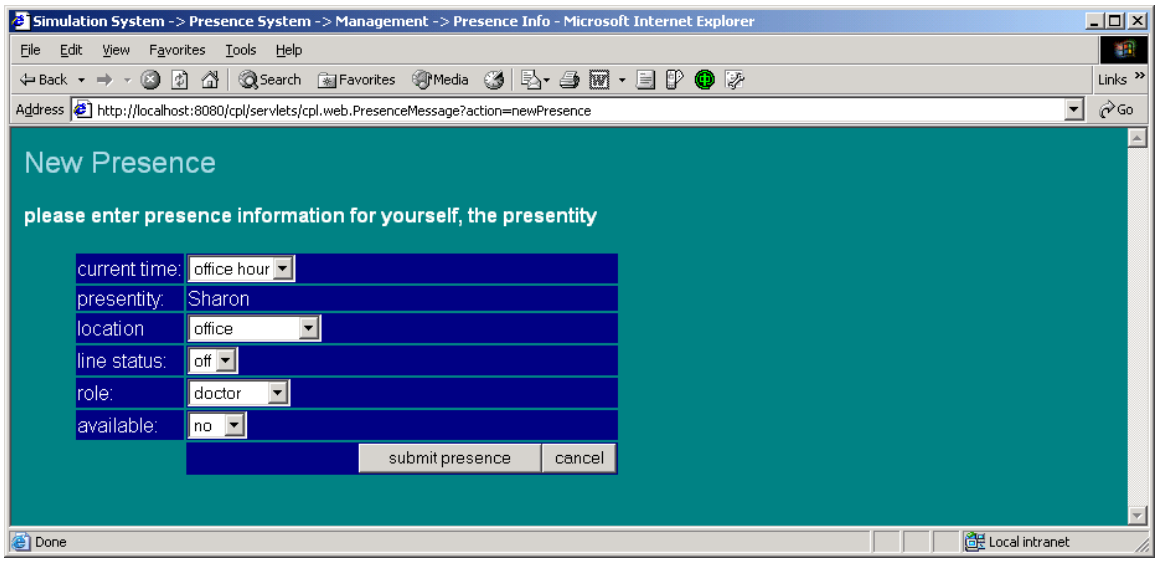

Figure 5.7 New Presence

(Sharon registers her presence information via the GUI)

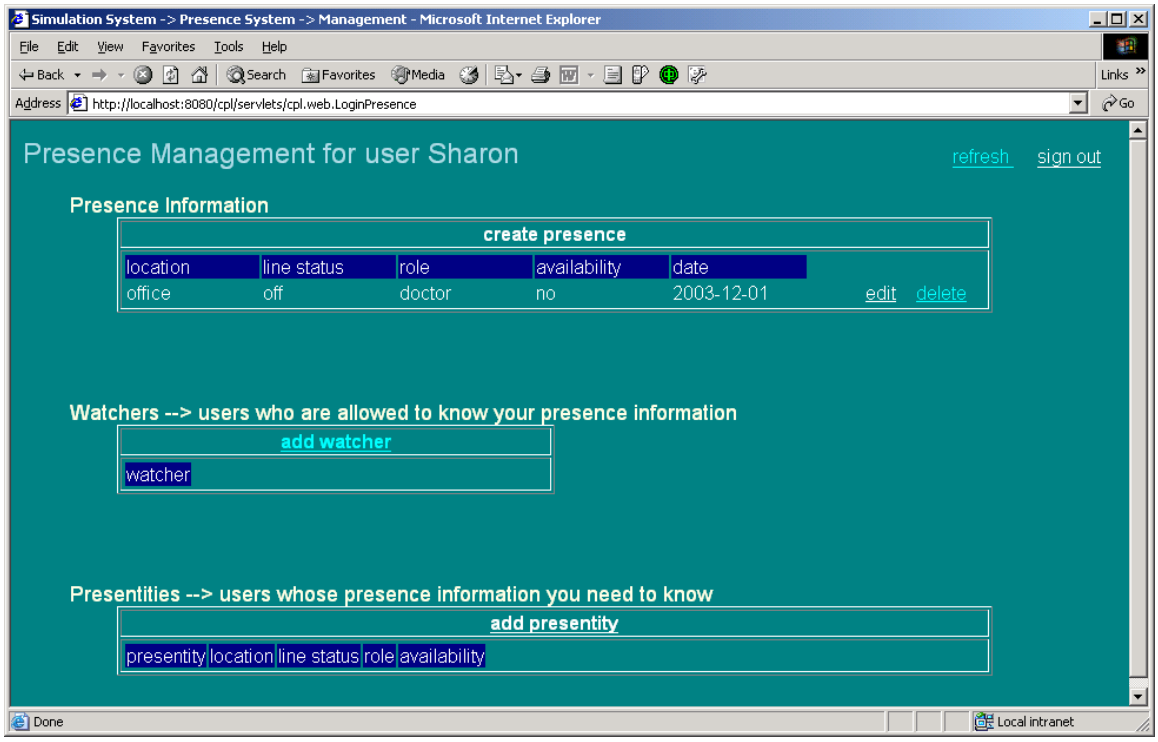

Figure 5.8 Presence Management Display
(Sharon's presence information has been registered)

Scenario 2: Sharon sends her subscription request to Stephen

In order to know Stephen's presence information, Sharon clicks the button "add presentity". After the presence system displays the GUI "Add Presentity", Sharon fills "Stephen" in the blank cell "Presentity" and submits the request (see Fig. 5.9). So Stephen has been added on Sharon's presentity list as shown in Fig. 5.10.

In Fig. 5.10, Stephen's presence information is filled by using data provided by Stephen. At this point, we assume that Stephen has no policies (see section 5.6) and only basic system services are provided in this aspect.

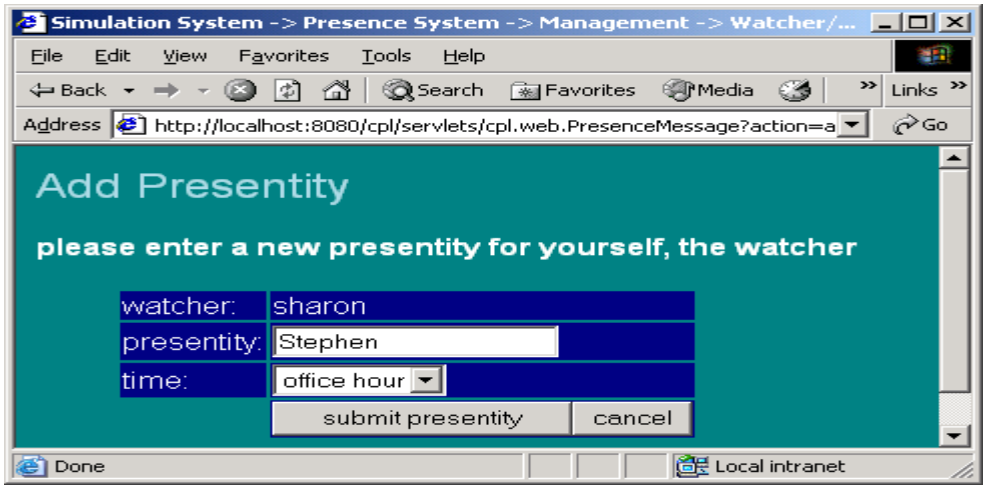

Figure 5.9 Add Presentity

(Sharon sends her subscription request to Stephen)

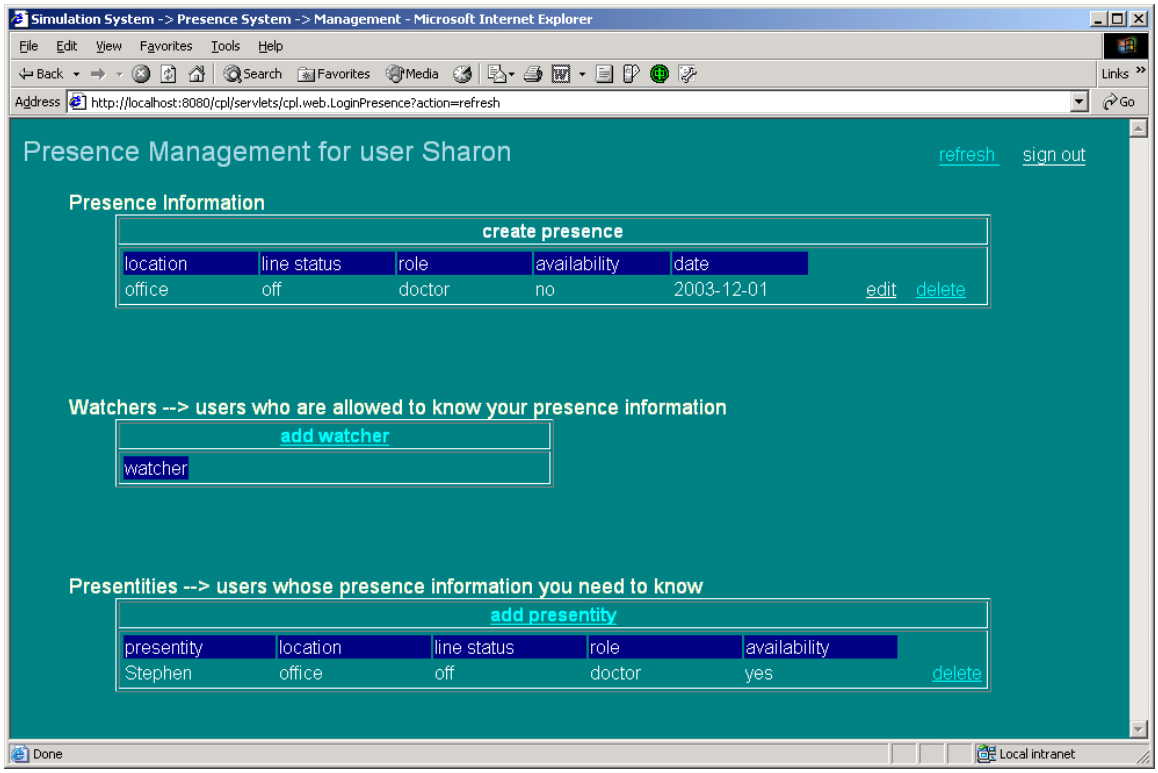

Figure 5.10 Presence Management Diaplay

(Stephen has been added to Sharon's presentity list)

Scenario 3: Sharon approves Edward's subscription request

To approve Edward's request, Sharon clicks the button "add watcher". The GUI "Add Watcher" is displayed. Sharon fills "Edward" in the blank cell "Watcher" as shown in Fig. 5.11.

After the request is submitted, Stephen is on Sharon's watcher list (see Fig. 5.12).

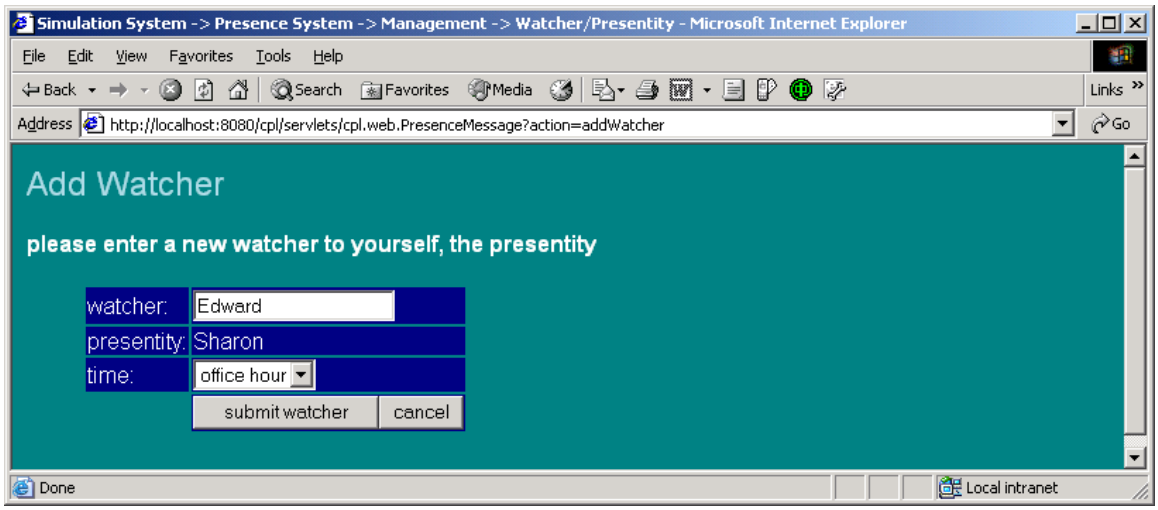

Figure 5.11 Add Watcher
(Sharon approves Edward's request)

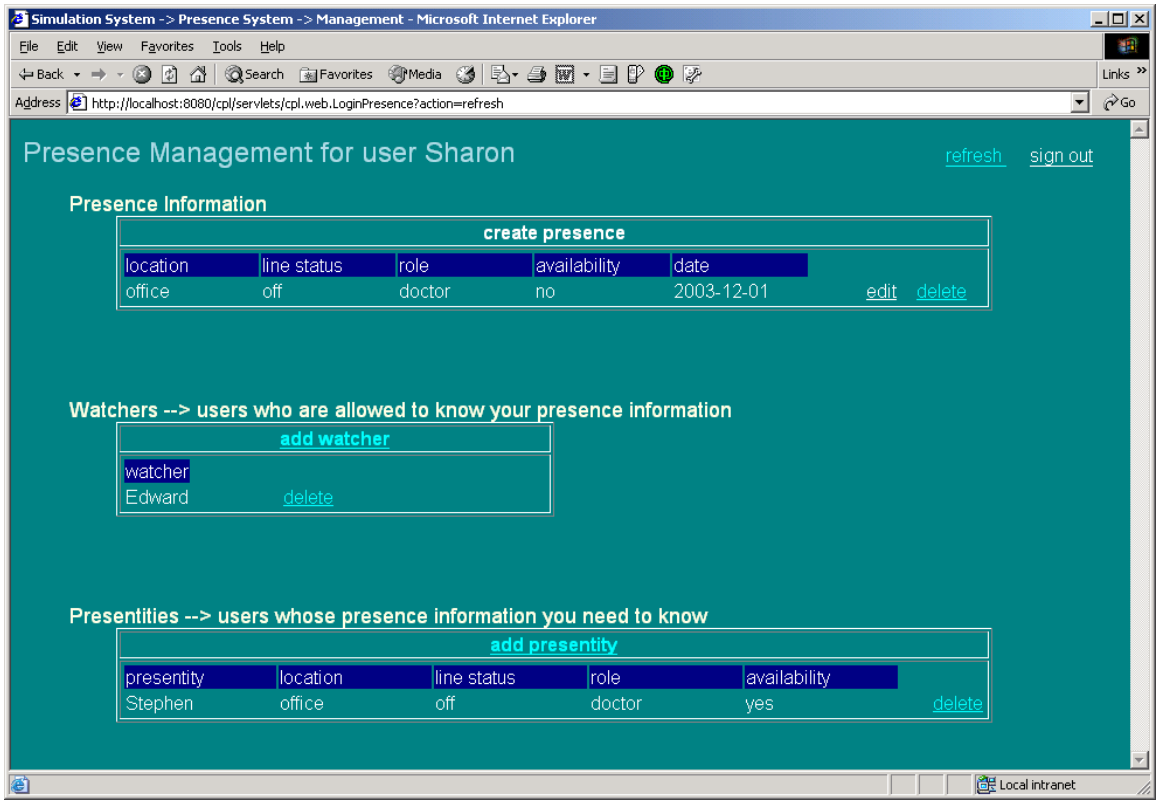

Figure 5.12 Presence Management Display
(Edward has been added to Sharon's watcher list)

Scenario 4: Sharon changes her presence status

To change her presence status, Sharon clicks the button "edit" for the presence information. The GUI "Update Presence" is displayed. Sharon changes her location from "office" to "meeting room", as shown in Fig. 5.13.

After the request is submitted, Sharon's presence information is updated (see Fig. 5.14).

In reality, we can expect that such updates will be performed automatically, by information received from sensors positioned in the environment.

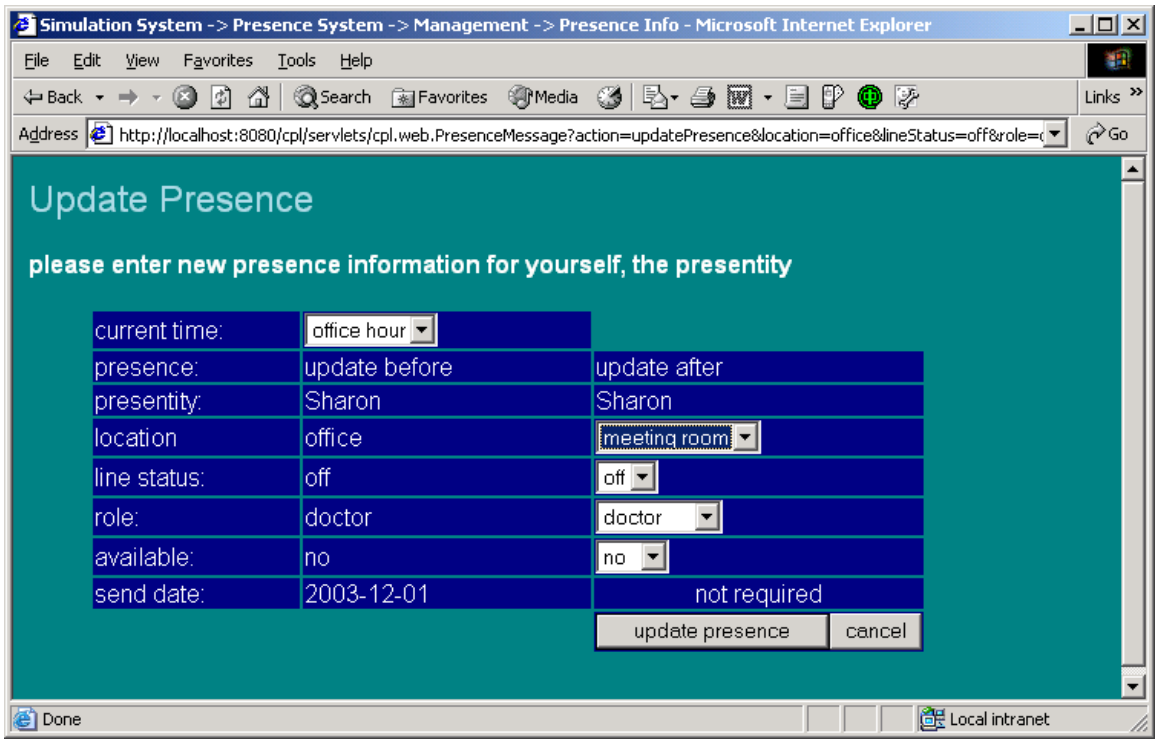

Figure 5.13 Update Presence

(Sharon's location is changed from "office" to "meeting room")

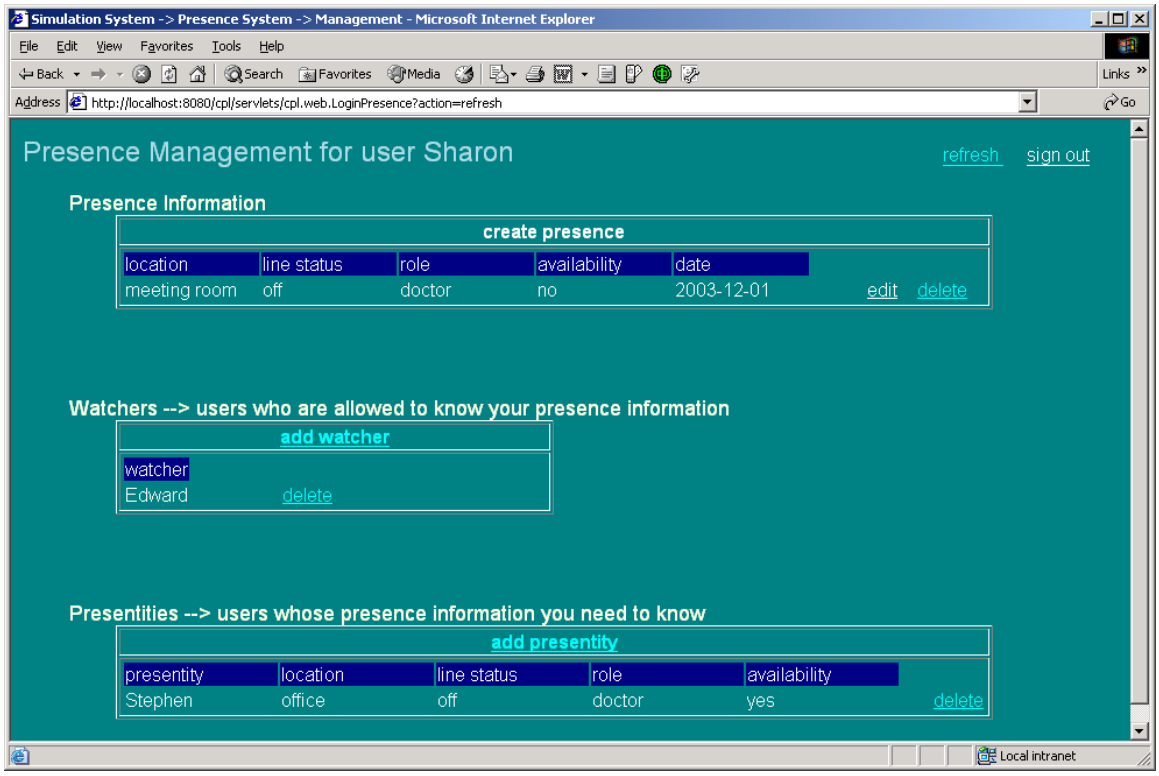

Figure 5.14 Presence Management Update
(Sharon's presence status displays she is in meeting room)

Sharon also can cancel her presence service relative to a presentity by clicking the button "delete" corresponding to the presentity. In a similar way, Sharon can terminate a service offered to a watcher by clicking the button "delete" corresponding to the watcher.

5.4 Policy System

5.4.1 Introduction

The policy system provides policy management services. A user can manage his phone account and his specific services (i.e. policies) in the policy system. He can create and modify his phone account; and he can create, modify and remove his policies.

A user has to register to the policy system to obtain an account before he can use the system. To get the account, the user Sharon, for example, needs a unique name, a unique logic phone (published phone) and a maximum of three physical phones. Each phone is identified by a unique URI. Sharon requests her phone account as shown in Fig. 5.15. Sharon's logic phone is identified by "sharon@example.com". She has three phone devices identified by "Sharon@dep1.example.com", "Sharon@site.uottawa.ca" and sharon@voicemail.example.com, respectively.

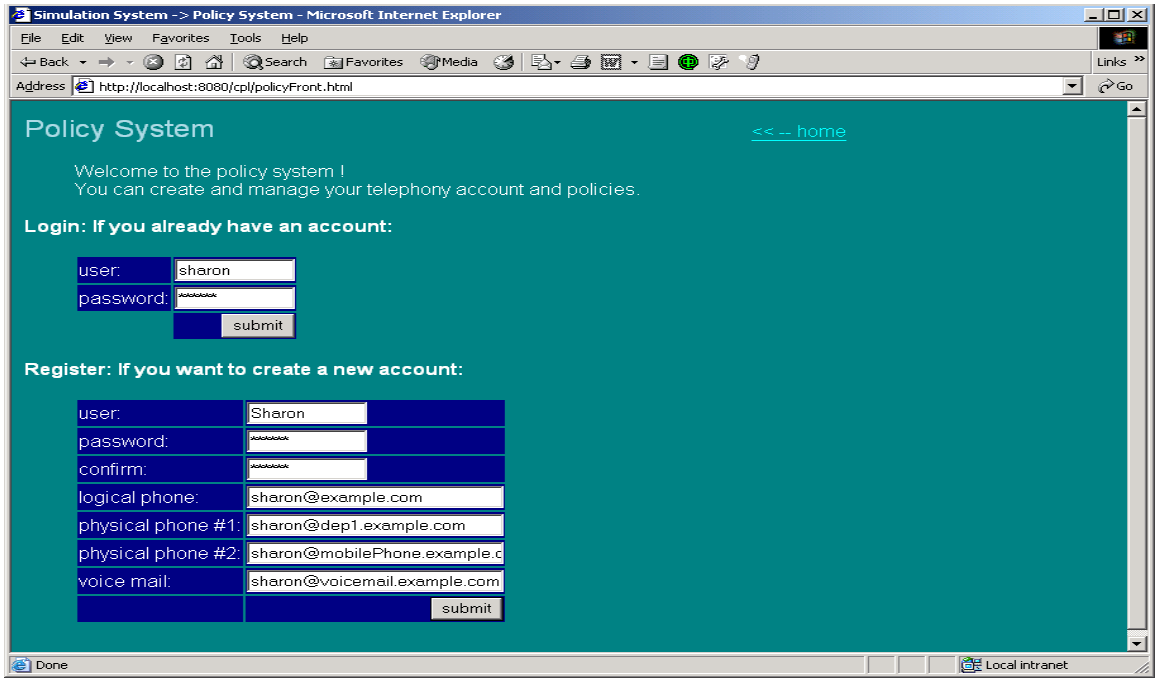

Figure 5.15 Policy System

(Sharon registers her personal information to the policy system)

After the registration, Sharon logins to the policy system and sees her phone account information as shown in Fig. 5.16.

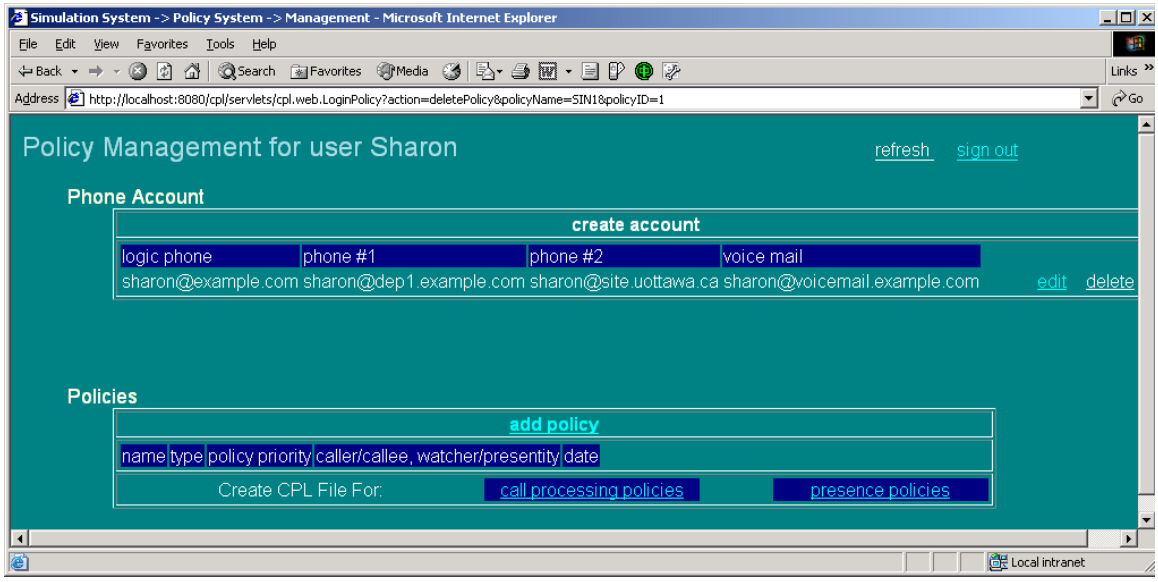

Figure 5.16 Policy Management

5.4.2 The Relationship between the GUI and Services

The policy system provides several services via the GUI “Policy Management”. In Fig. 5.17, each button on the GUI corresponds to a service in the system. The relationship between the buttons and the services is shown in Table. 5.2.

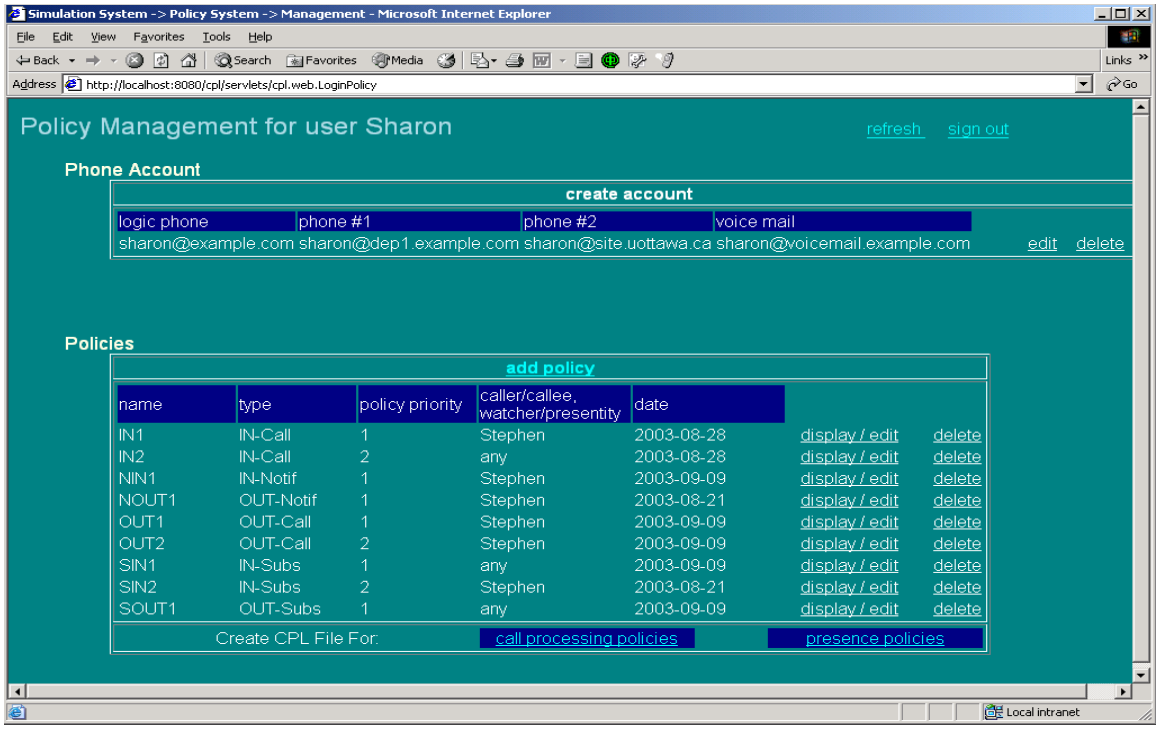

Figure 5.17 Policy Management with Six Types of Policies

GUI Button	Policy Service
Personal Account	
Create account	Register the user to the system
Edit	Update the user's phone account in the system
Delete	Remove the user from the system
Policies	
Add policy	Specify a new policy by the user
Display	Display the policy
Edit	Modify the policy
Delete	Remove the policy
Create CPL files	
Call processing policy	Save a file in extended CPL to describe call processing services
Presence policy	Save a file in extended CPL to describe presence services

Table 5.2 Relationship between the GUI and Policy Services

5.4.3 Policy Management

There are six types of policies in the policy system as listed in Table 5.3. An end user can have a maximum of six types of policies; each type may contain multiple policies. There may be some conflicts among these policies inside one type. For example, Stephen likes to forward his incoming calls to his voice mail from 9:00 am to 10:00 am every morning on workdays. Stephen likes to take his wife's calls unconditionally. The two incoming call policies conflict if Stephen's wife calls at 9:30. In order to solve such conflicts, the simulation system gives a number to each policy in each type. The policy with the highest priority is numbered "1". The policy with the second highest priority is numbered "2" and so on (see Fig. 5.12).

Action Tag in CPL	**Policy Type Name**	**Explanation**
incoming	IN-Call	Deal with incoming call
outgoing	OUT-Call	Deal with outgoing call
incoming subscription	IN-Subs	Deal with incoming subscription request
outgoing subscription	OUT-Subs	Deal with outgoing subscription request
incoming notification	IN-Notif	Deal with incoming notification
Outgoing notification	OUT-Notif	Deal with outgoing notification

Table 5.3 Policy Type

In the processing of one type service, the policies are checked in the order of priority within the type. The highest priority policy is checked first. As soon as a policy matches the criteria, the remaining policies are ignored. If Stephen gives a higher priority to the policy for his wife's calls, he can take his wife's calls at 9:30 am. The forwarding policy with the lower priority is ignored in this case. Usually specific policies will be given higher priority than more general ones. A system for detecting CPL policy conflicts was studied in [24]. The research on feature interactions in traditional telecommunications systems can be found in [25] [26].

5.4.4 Policy Services

There are two main types of services: personal account management and policy management.

In the policy system, most services are provided via the GUI "Policy Management" except for account creation, which is offered via the GUI "Policy System". Several main services of the system are illustrated in the following scenarios for user Sharon:

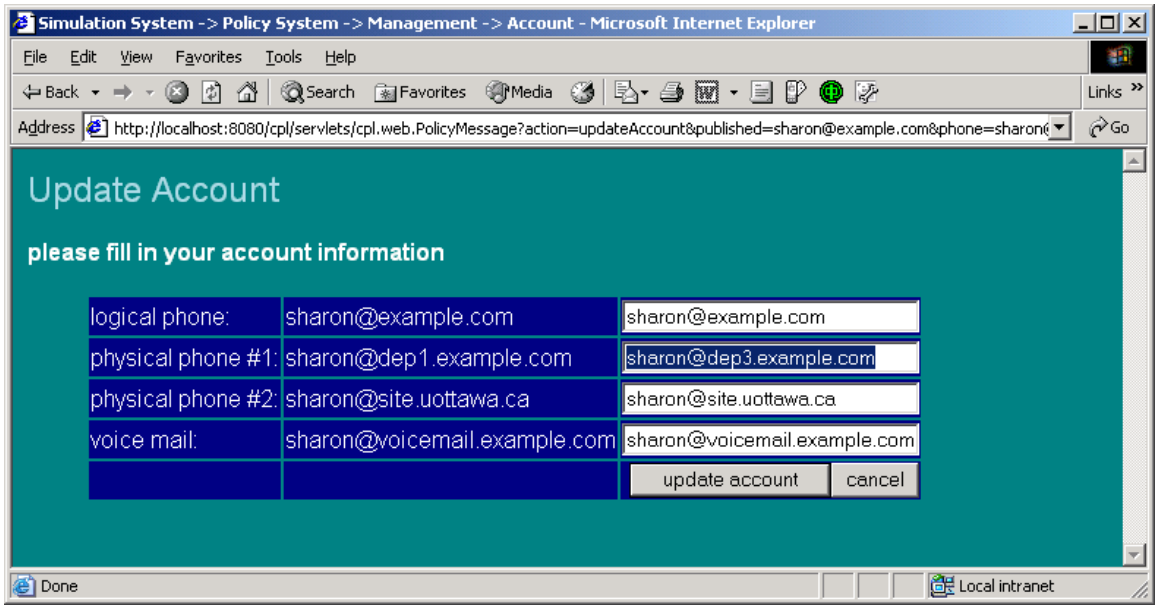

Figure 5.18 Update Account
(Sharon modifies her phone account information)

Scenario 1: Sharon modifies her phone account

Sharon clicks button "edit" for her phone account, the system displays the GUI "Update Account". Referencing her original account information, Sharon changes her physical phone#1 from "sharon@dep1.example.com" to "sharon@dep3.example.com", as shown in Fig. 5.18. After the request "Update Account" is submitted, Sharon's account has been updated (see Fig. 5.19).

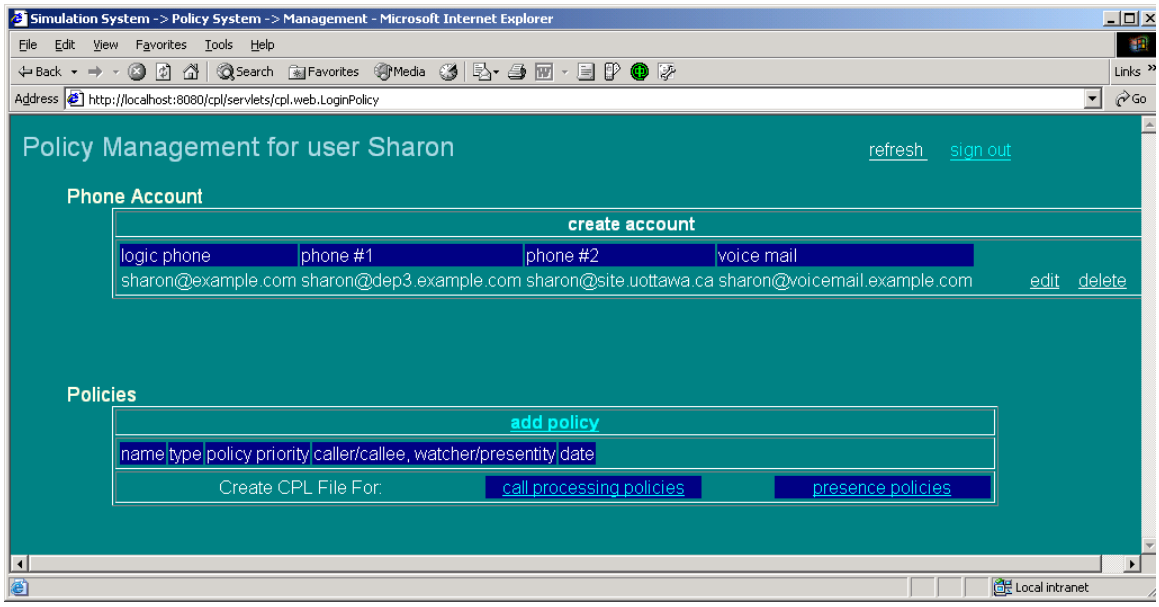

Figure 5.19 Policy Management Update
(Sharon's phone#1 is changed to "Sharon@dep3.example.com")

Scenario 2: Sharon creates her policies

Sharon clicks the button "add policy", the system displays the GUI "Add Policy" (see Fig. 5.20).

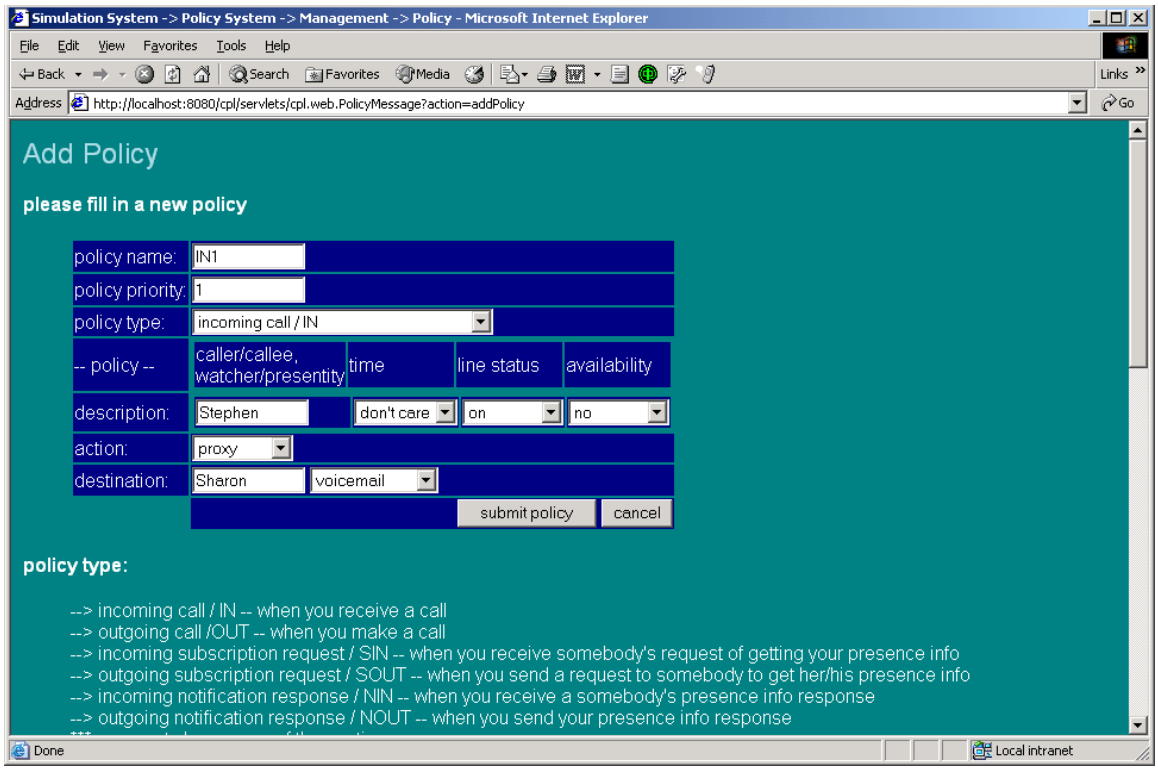

Figure 5.20 Add Policy
(Sharon creates her forwarding policy for incoming calls)

Sharon would like to have a service (policy) to forward all her incoming calls to her voice mail when she is talking on her phone, unwilling to answer other phone calls. Sharon specifies her policy as follows:

Policy type: incoming call/IN
Policy priority: 1
Policy name: IN1
Caller: Stephen
Time: don't care
Line Status: on
Availability: no
Action: proxy
Destination: Sharon's voice mail

This service is based on Sharon's presence status. According to the policy, Stephen's incoming calls will be forwarded to Sharon's voice mail when she is talking on her phone unwilling to answer other phone calls. The instructions displayed on the GUI "Add Policy" provide online help to users for specifying their policies correctly.

After Sharon's incoming call handling policy#1 is submitted, the policy "IN1" has been added into her policy list (see Fig. 5.21).

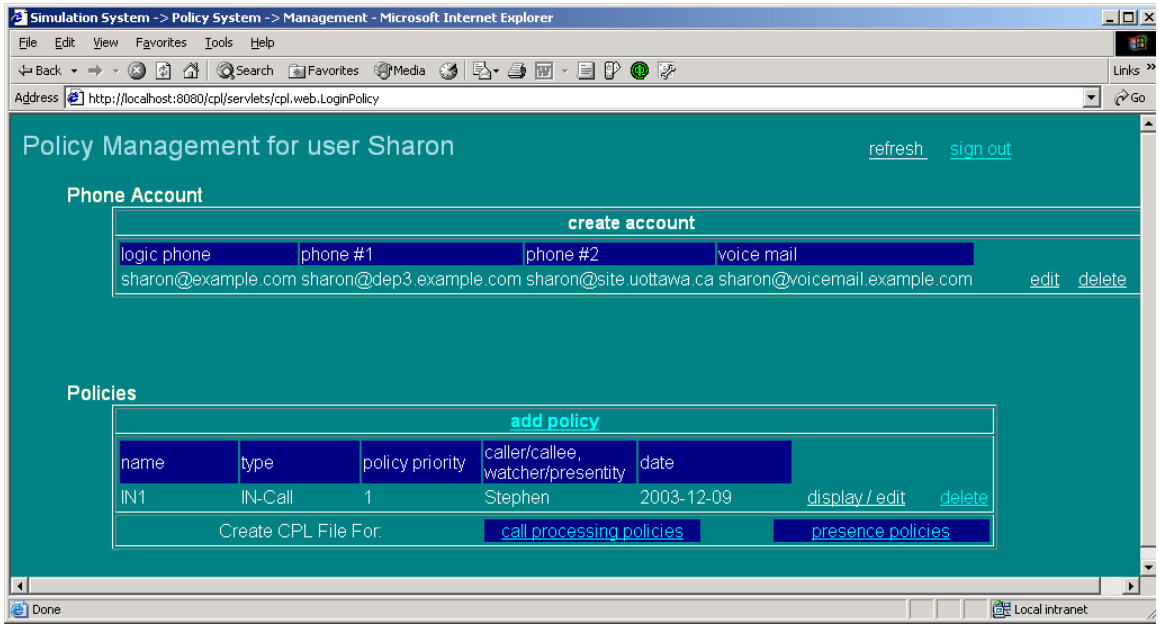

Figure 5.21 Policy Management Display
(Sharon's incoming call policy "IN1" has been created successfully)

Scenario 3: Sharon modifies her policies

Sharon clicks the button "display/edit" for policy "IN1", the system displays the GUI "Update Policy" as shown in Fig. 5.22. Sharon changes caller "Stephen" to "Edward" in the policy "IN1". After Sharon submits the policy, she sees that the policy "IN1" has been updated (see Fig. 5.23).

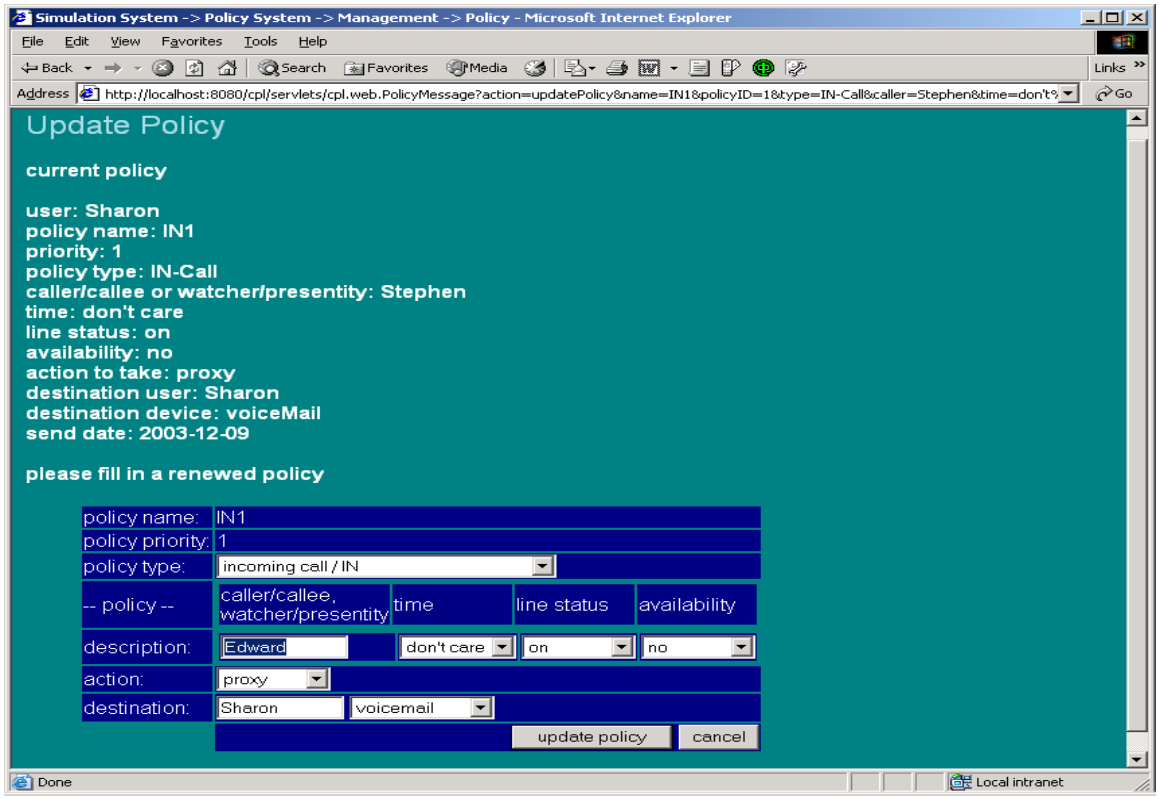

Figure 5.22 Update Policy

(Sharon changes the caller “Stephen” to “Edward”)

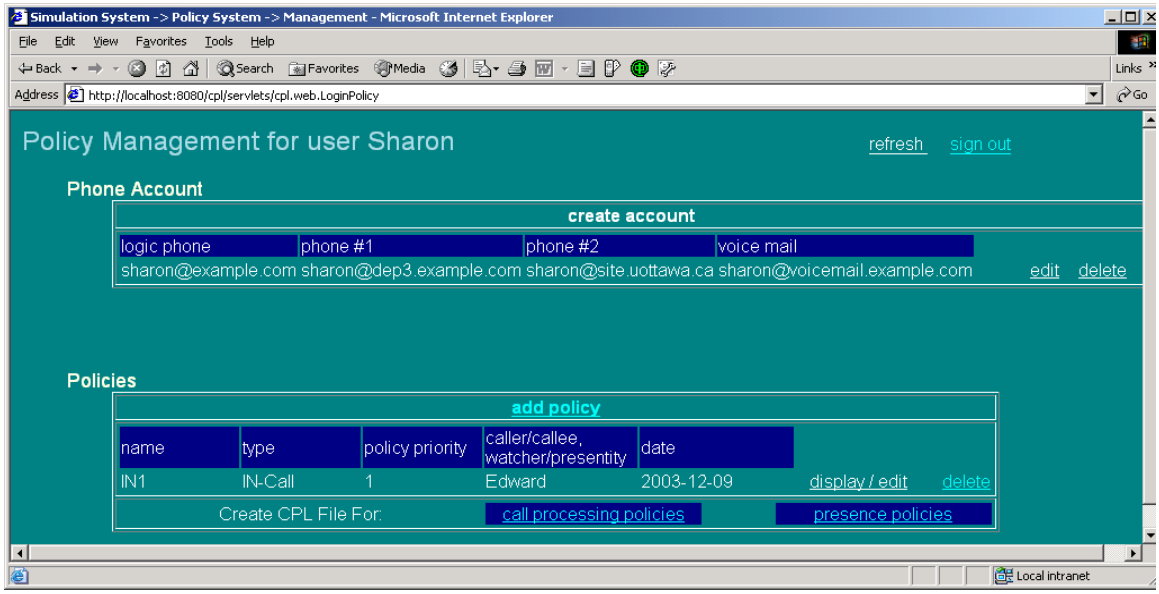

Figure 5.23 Policy Management Update

(The caller is changed to “Edward” in Sharon’s policy “IN1”)

Scenario 4: Sharon saves her policies in Extended CPL as **files**

The system can translate end user's services into CPL and CPL extensions and then save them in files for the users. A service system, either the presence system or the call processing system, can recognize and provide these services specified in Extended CPL.

Sharon clicks the button "call processing policies", the system saves Sharon's policies in type "IN" and "OUT" into a file named "Sharon-c.cpl" in the system. In the file, these policies are ordered by policy type.

If Sharon clicks the button "presence policies", the system saves Sharon's policies in types "IN-Subs", "OUT-Subs", "IN-Notif" and "OUT-Notif" into a file named "Sharon-p.cpl" in the system. These policies are ordered by policy type in the file.

In conclusion, the policy system offers an end user the following operations:

1). Create a phone account for a new user;
2). Create policies in six types for an authorized user;
3). Add, remove, modify and display policies for an authorized user;
4). Create and save a file in Extended CPL to describe presence services;
5). Create and save a file in Extended CPL to describe call processing services.

5.5 Call Processing System

The call processing system offers call processing services to users. The home GUI "Call Processing System" is shown in Fig. 5.24.

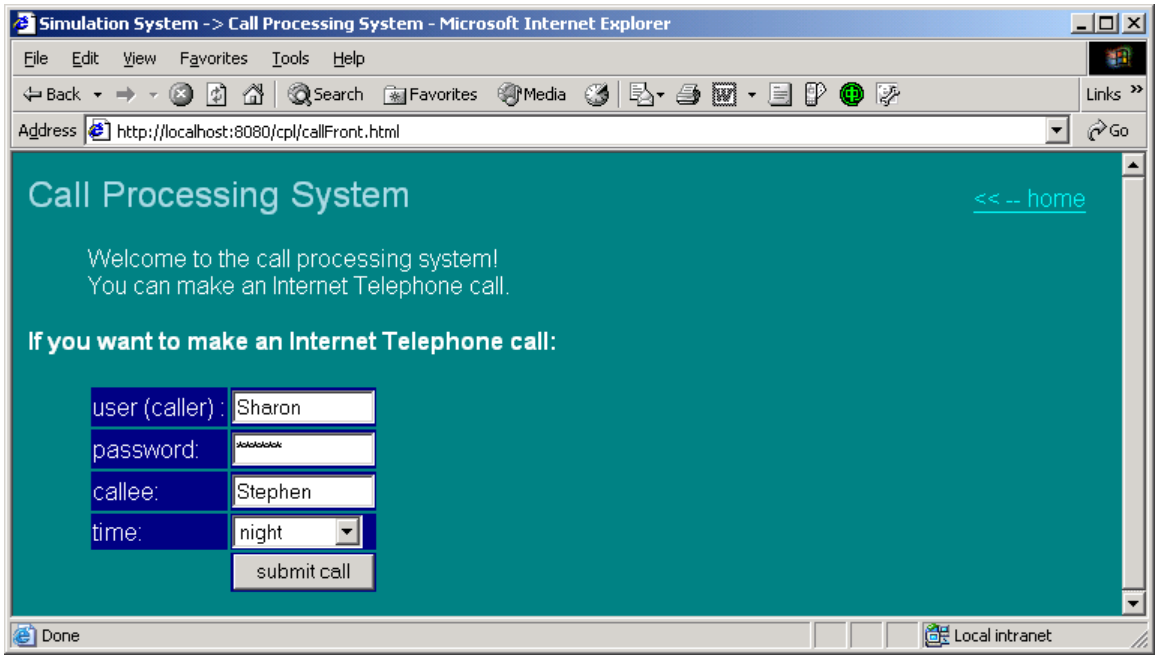

Figure 5.24 Call Processing System

In the call processing system, a registered user can make a phone call to another registered user. For example, user Sharon likes to make a call to Stephen. She fills her name, password, callee (Stephen) in her request form and submits her call as shown in Fig. 5.24. As a system default service, her call to Stephen is connected with no policies applied (see Fig. 5.25).

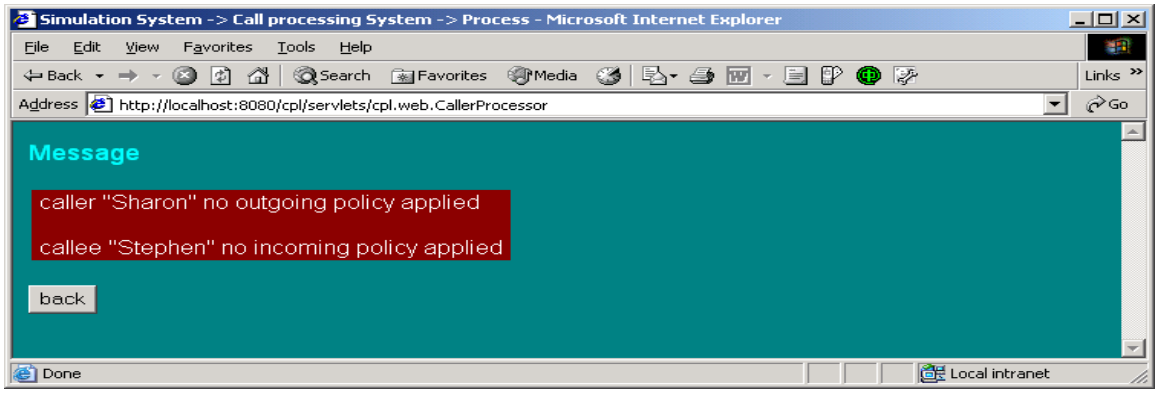

Figure 5.25 Message
(Sharon's call is connected without any policies applied)

5.6 End User Services

The simulation system offers both basic services and end user specific service. Basic presence services and basic call processing services have been described in section 5.3 and section 5.5 respectively. This section will illustrate the end user's specific services.

5.6.1 Sharon's Specific Services

Suppose that Sharon works at "example.com" in New York, U.S.A. She is an end user of the simulation system. She would like to have the presence services and call processing services listed in Table 5.4.

Service (Policy)	Name	Description
Presence Policy		
Outgoing Subscription Blocking #1	SOUT1	Blocks outgoing subscription requests outside of work hours.
Incoming Subscription Rejection #1	SIN1	Rejects Stephen's incoming subscription requests if Sharon is on the phone
Outgoing Notification Controlling #1	NOUT1	Blocks notifications to Stephen if Sharon is on her phone
Call processing Policy		
Outgoing Call Blocking #1	OUT1	Blocks calls to Stephen when Sharon does not like to communicate with others during work hours.
Outgoing Call Forwarding #2	OUT2	Her calls to Stephen are automatically forwarded to Stephen's cell phone in lunchtime.
Incoming Call Rejection #1	IN1	Rejects incoming calls from Stephen at night.
Incoming Call Forwarding #2	IN2	Forward incoming calls to her voice mail if she is talking on her phone while her availability status is "no".

Table 5.4 Sharon's Policies

How to create a policy has been described by scenario 2 in section 5.4. Sharon repeats the same procedure in scenario 2 for each policy application. After all these policies are complete, the summary of Sharon's policies is displayed in Fig. 5.26. These policies are ordered by policy type. Inside each type, policies are ordered by policy numbers: the smaller the policy number, the higher the priority.

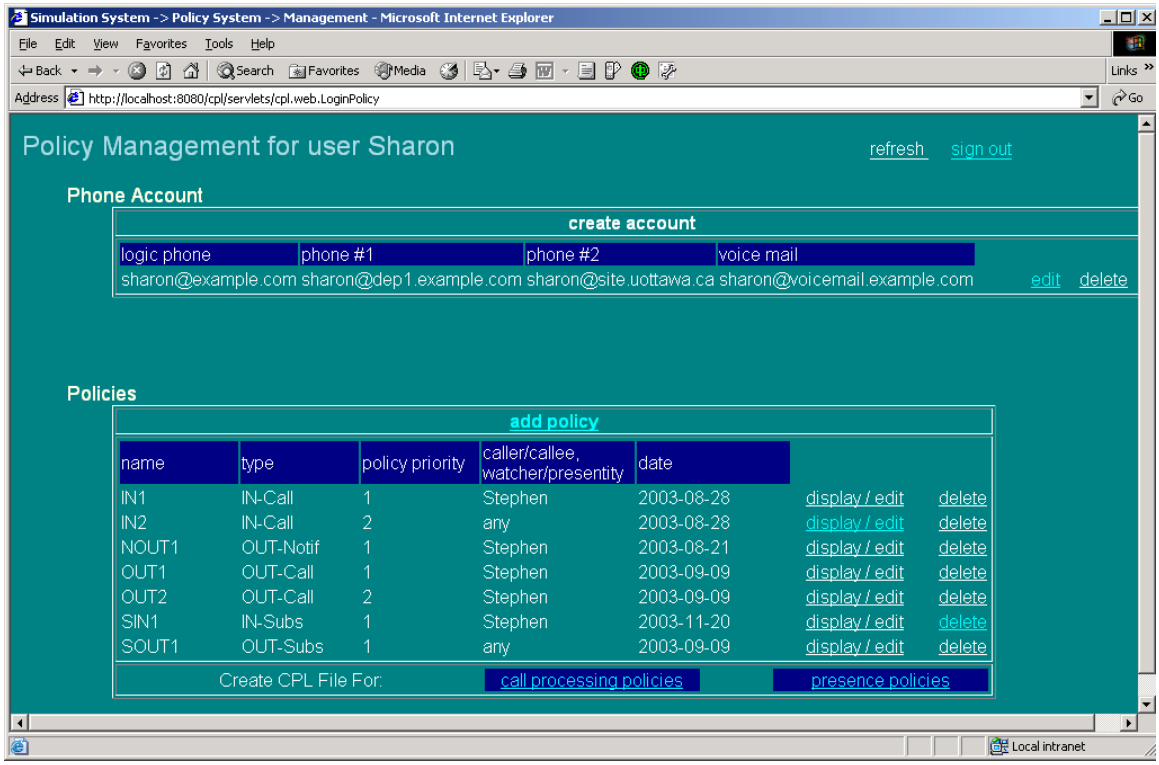

Figure 5.26 Policy Management Display
(Sharon has created seven policies)

5.6.2 Sharon's Services Specified in CPL

The scenario 4 in section 5.4 describes how the simulation system saves a user's services in extended CPL into files. Sharon just follows the procedures in scenario 4 and gets two separate files with extension "cpl". The File "Sharon-p.cpl" contains Sharon's presence services specified in Extended CPL (see Fig. 5.27). The file "Sharon-c.cpl" contains Sharon's call processing services specified in Extended CPL (see Fig. 5.28). We assume that CPL is defined in the default name space that can be accessed at the URI

"http://www.site.uottawa.ca/~djiang/cpl.dtd"; and CPL extensions for presence are defined in the name space "cplPresence" which can be accessed at the URI "http://www.site.uottawa.ca/~djiang/cplPresence.dtd".

```
<?xml version="1.0" ?>
<cpl xmls="http://www.site.uottawa.ca/~djiang/cpl.dtd"
 xmls:cplPresence="http://www.site.uottawa.ca/~djiang/cplPresence.dtd">

<!-- ************  Incoming Subsciption Policies  ************ -->
<cplPresence: incoming subscription>
  <!-- ----  Policy # 1 - SIN1  ------ -->
  <address-switch field = "orgin">
  <address is = "sip:stephen@example.com">
    <cplPresence:presence-switch presentity="sip:sharon@example.com">
    <cplPresence:presence lineStatus = "on" >
    <cplPresence:success>
      <reject/>
    </cplPresence:success>
    </cplPresence:presence>
    </cplPresence:presence-switch >
  </address >
  </address-switch >
</cplPresence:incoming subscription>


<!-- ************  Outgoing Subscription Policies  ************ -->
<cplPresence:outgoing subscription>
  <!-- ----  Policy # 1 -- SOUT1  ------ -->
    <time-switch tzid="America/New-York"
                 tzurl="http://example.com/tz/America/New_york">
    <time dtstart="20030101T180000" duration = "PT14H"  freq = "weekly"
          byday = "MO, TU, WE, TH, FR">
      <reject/>
    </time>
    </time-switch >
</cplPresence:outgoing subscription>


<!-- ************  Outgoing Notification Policies  ************ -->
<cplPresence:outgoing notification>
  <!-- ----  Policy # 1 -- NOUT1  ------ -->
  <address-switch field = "orgin">
  <address is = "sip:stephen@example.com">
      <cplPresence:presence-switch
                   presentity="sip:sharon@example.com">
      <cplPresence:presence lineStatus = "on" >
      <cplPresence:success>
        <reject/>
      </cplPresence:success>
      </cplPresence:presence>
      </cplPresence:presence-switch >
  </address >
  </address-switch >
</presence: outgoing notification>
</cpl>
```

Figure 5.27 File "Sharon-p.cpl"

```
<?xml version="1.0" ?>
<cpl xmls="http://www.site.uottawa.ca/~djiang/cpl.dtd"
 xmls:cplPresence="http://www.site.uottawa.ca/~djiang/cplPresence.dtd">


<!-- ************  Incoming Call Policies  ************ -->
<incoming>
  <!-- ----  Policy # 1 -- IN1  ------ -->
  <address-switch field = "orgin">
  <address is = "sip:stephen@example.com">
    <time-switch tzid="America/New-York"
                 tzurl = "http://example.com/tz/America/New_york">
    <time dtstart="20030101T180000" duration = "PT14H"  freq = "weekly"
          byday = "MO, TU, WE, TH, FR">
      <reject/>
    </time>
    </time-switch >
  </address >
  </address-switch >

  <!-- ----  Policy # 2 -- IN2  ------ -->
    <time-switch tzid="America/New-York"
                 tzurl = "http://example.com/tz/America/New_york">
    <time dtstart="20030101T090000" duration = "PT8H"  freq = "weekly"
          byday = "MO, TU, WE, TH, FR">
      <cplPresence:presence-switch
                   presentity="sip:sharon@example.com">
      <cplPresence:presence lineStatus = "on" availability = "no">
      <cplPresence:success >
        <location url = "sip:sharon@voicemail.example.com">
          <proxy/>
        </location>
      </cplPresence:success>
      </cplPresence:presence>
      </cplPresence:presence-switch >
    </time>
    </time-switch >
</incoming>


<!-- ************  Outgoing Call Policies  ************ -->
<outgoing>
  <!-- ----  Policy # 1 -- OUT1  ------ -->
  <address-switch field = "orgin">
  <address is = "sip:stephen@example.com">
    <time-switch tzid="America/New-York"
                 tzurl = "http://example.com/tz/America/New_york">
    <time dtstart="20030101T090000" duration = "PT8H"  freq = "weekly"
          byday = "MO, TU, WE, TH, FR">
      <cplPresence:presence-switch
                   presentity="sip:sharon@example.com">
      <cplPresence:presence availability = "no" >
      <cplPresence:success >
        <reject/>
      </cplPresence:success>
      </cplPresence:presence>
      </cplPresence:presence-switch >
    </time>
    </time-switch >
  </address >
  </address-switch >
```

```
  <!-- ----  Policy # 2 -- OUT2  ------ -->
  <address-switch field = "orgin">
  <address is = "sip:stephen@example.com">
    <time-switch tzid="America/New-York"
                 tzurl = "http://example.com/tz/America/New_york">
    <time dtstart="20030101T120000" duration = "PT1H"  freq = "weekly"
          byday = "MO, TU, WE, TH, FR">
      <cplPresence:presence-switch
                   presentity="sip:sharon@example.com">
      <cplPresence:presence lineStatus = "off" >
      <cplPresence:success >
        <location url = "sip:stephen@mobilePhone.example.co">
          <proxy/>
        </location>
      </cplPresence:success>
      </cplPresence:presence>
      </cplPresence:presence-switch >
    </time>
    </time-switch >
  </address >
  </address-switch >
</outgoing>

</cpl>
```

Figure 5.28 File “Sharon-c.cpl”

5.6.3 Use Case Tests for Sharon’s Policies

In this section, the simulation system will test if it can offer Sharon’s services according to Sharon’s policies. Our test cases are chosen as examples and do not try to be exhaustive. Four test cases are shown as follows:

Case 1: SOUT1 – Outgoing Subscription Blocking

The policy “SOUT1” is that Sharon blocks all outgoing subscription requests outside of work hours.

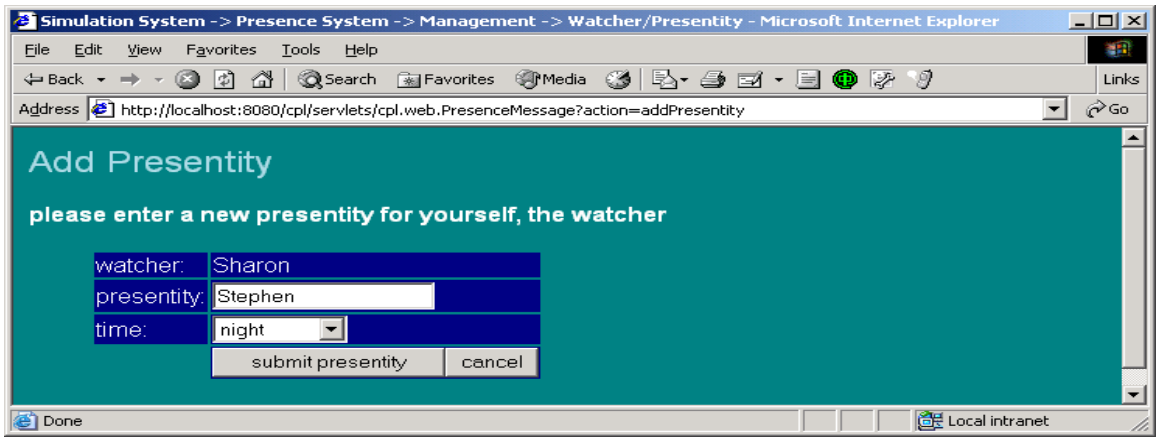

Figure 5.29 Add Presentity for Testing Policy "SOUT1"
(Sharon sends a subscription request to Stephen at night)

Sharon sends a subscription request to Stephen at night, as shown in Fig. 5.29. Her request result is shown in Fig.5.30. Sharon's subscription request is rejected according to her outgoing subscription policy "SOUT1".

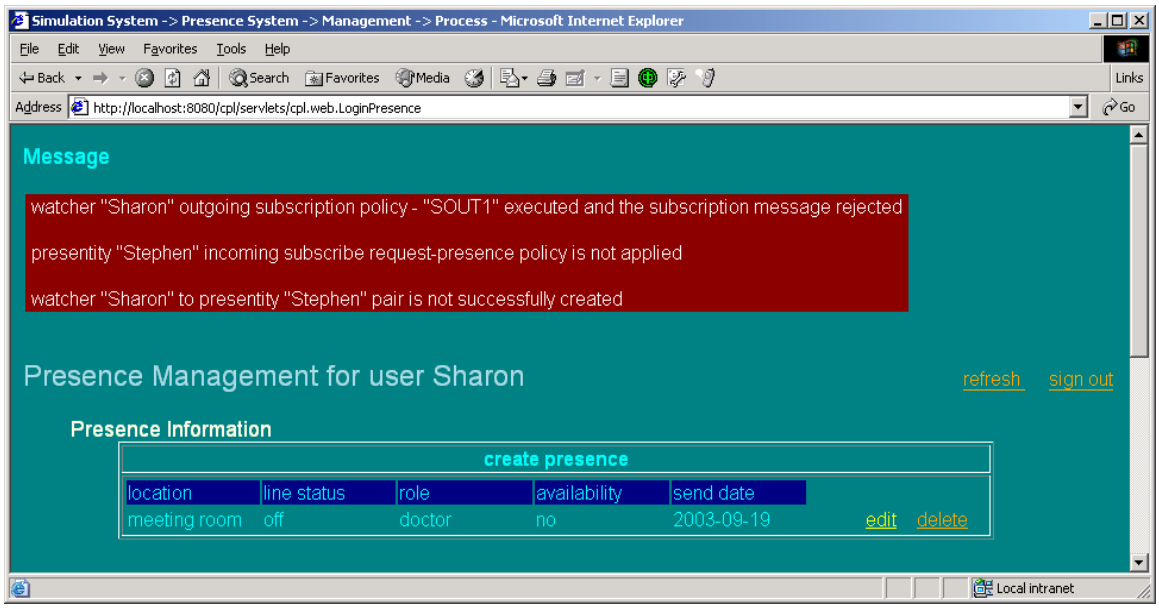

Figure 5.30 Message for "SOUT1" Test Result
(Sharon's request is rejected according to her policy "SOUT1")

Test case 2: NOUT1 – Notification controlling

The policy "NOUT1" is that Sharon blocks notifications to Stephen when she is on the phone.

Stephen is a watcher of Sharon. During work hours, Sharon changes her line status from "off" to "on", when she picks up her phone to make a call to somebody (see Fig. 5.31). The event processing results are shown in Fig. 5.32. When Sharon's line status is changed from status "off" to status "on", Stephen is not notified. Sharon prevents Stephen from knowing that she is on her phone, according to her policy "NOUT1".

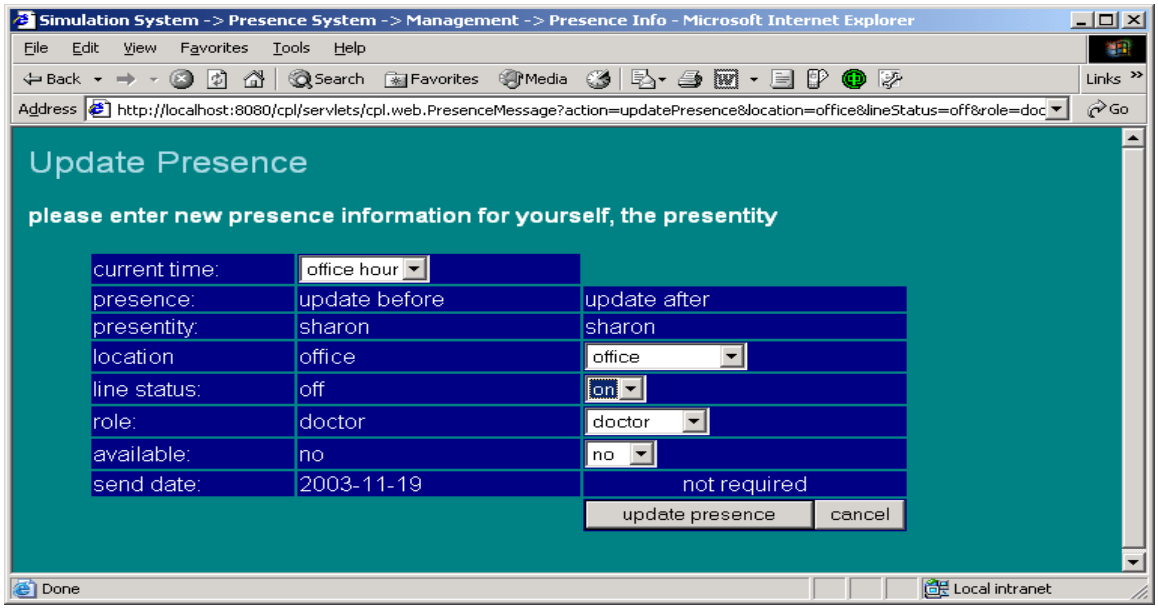

Figure 5.31 Update Presence for Testing Policy "NOUT1"
(Sharon makes a phone call i.e. line status is changed from "off" to "on")

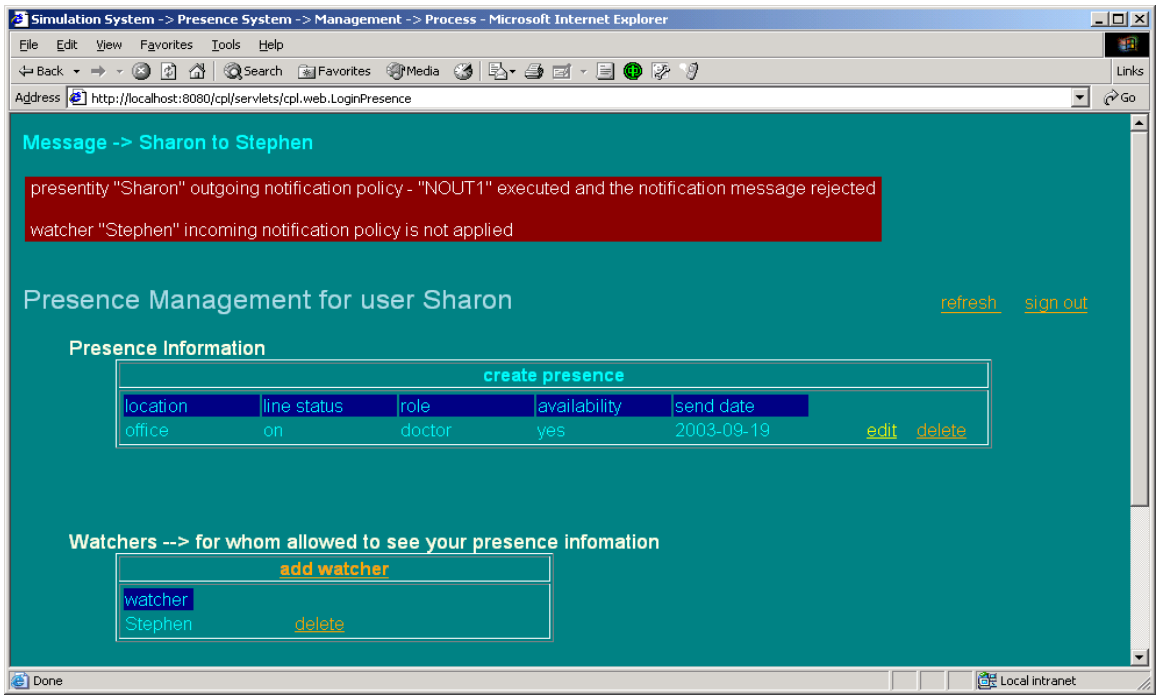

Figure 5.32 Message for "NOUT1" Test Result
(Stephen is not notified when Sharon makes a phone call, according to Sharon's policy NOUT1)

Test case 3: OUT1 - Screening outgoing call

The policy "OUT1" is that Sharon forbids herself from calling Stephen when her availability status is "no" during work hours. This policy can be treated as a self-imposed restriction.

When Sharon's availability status is "no" in work hours, she submits a call to Stephen, as shown in Fig. 5.33. Her call processing result is shown in Fig. 5.34. Her call is rejected according to her outgoing call policy "OUT1".

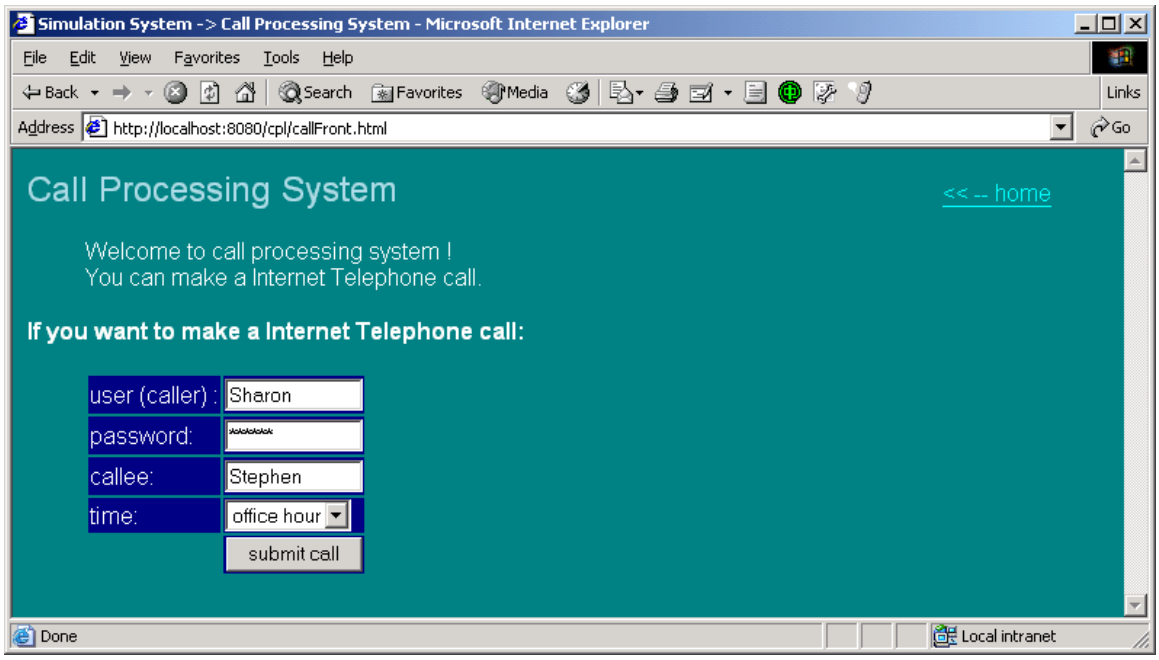

Figure 5.33 Call Processing System for Testing Policy "OUT1"
(Sharon makes a call to Stephen in office hour)

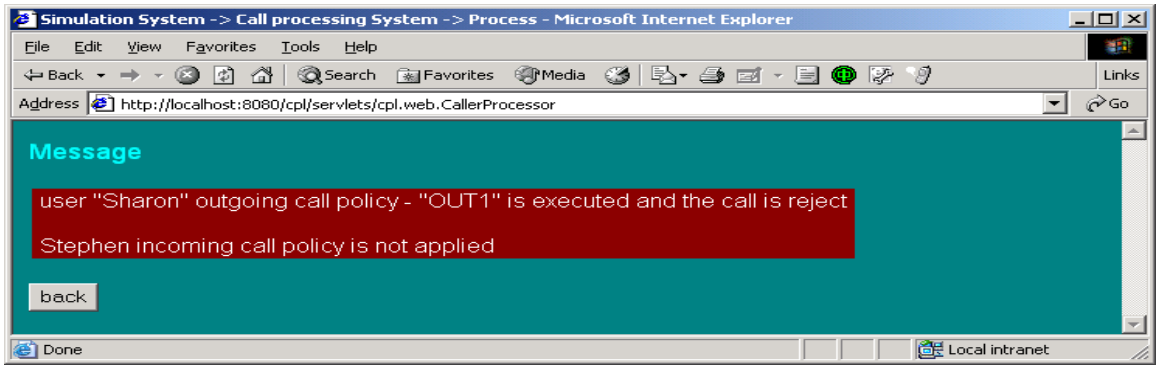

Figure 5.34 Message for "OUT1" Test Result
(Sharon's call is rejected according to her policy "OUT1")

Test case 4: IN2 - Forwarding incoming call

The policy "IN2" is that Sharon forwards all incoming calls to her voice mail if she is talking on her phone with her availability status "no".

When Sharon is talking on her phone, unwilling to take other phone calls, Stephen makes a call to Sharon, as shown in Fig. 5.35. Stephen's call processing result is shown in Fig. 5.36. Stephen's call is forwarded to Sharon's voice according to Sharon's policy "IN2".

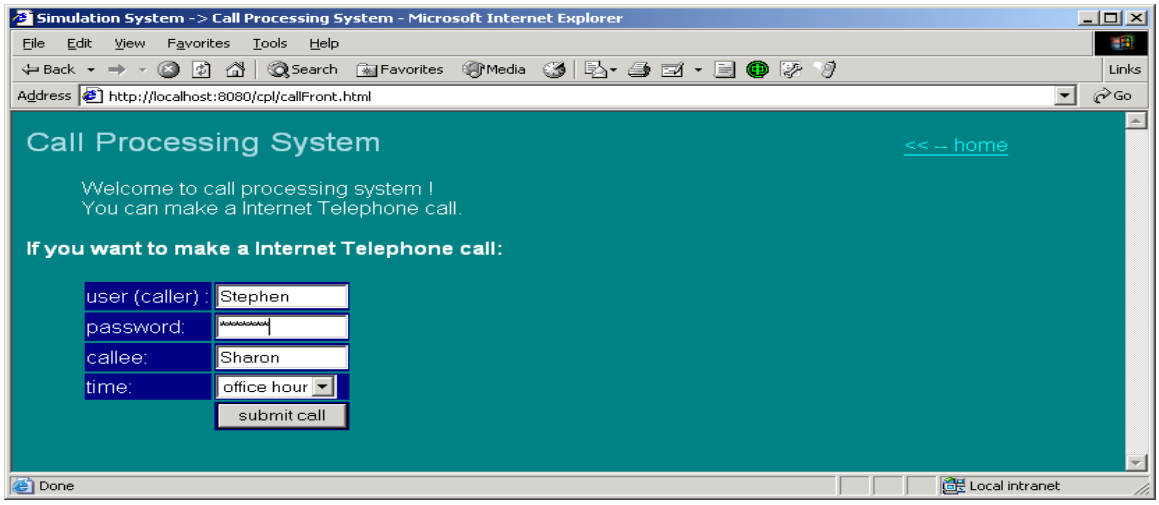

Figure 5.35 Call Processing System for Testing Policy "IN2"
(Stephen makes a call Sharon)

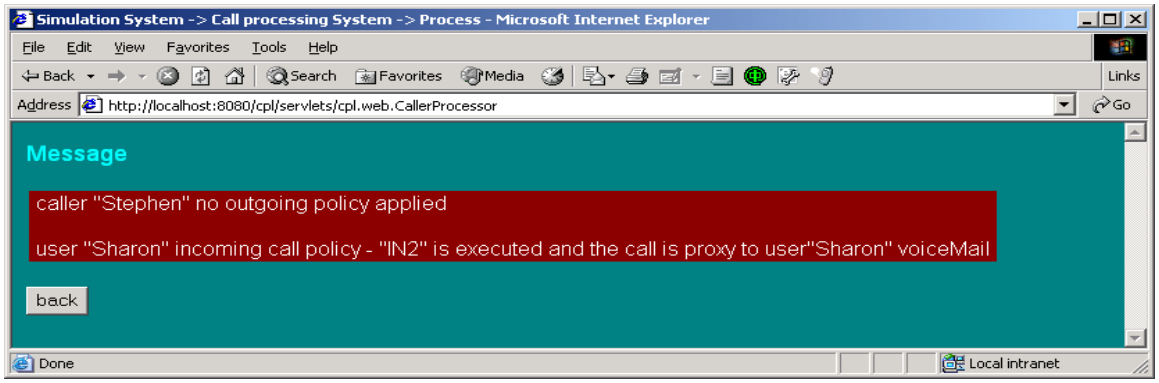

Figure 5.36 Message for "IN2" Test Result
(Stephen's call is forwarded to Sharon's voice mail according to Sharon's policy "IN2")

5.7 Conclusion

This chapter introduces and describes a simulation environment of Internet Telephony services specified in CPL and CPL extensions. The system design ideas and the system structure have been discussed in the chapter. Three subsystems are described individually: the presence system, the call processing system and the policy system. Basic services, either in the presence system or in the call processing system, are described through system service scenarios. End user's specific services are illustrated and tested in the simulation system.

The simulation system presented in this chapter shows the results that could be obtained by fully implemented presence system and call processing system, which would require much more work. Our own main contribution i.e. the CPL extensions discussed in chapter 4, were fully implemented, and the extensions of presence information, excluding the parameter "role" (section 4.3.1) were included in the simulation system.

Chapter 6 Summary and Future Work

This chapter reviews the contributions of the thesis and discusses the further work related to the topic.

6.1 Contributions of the Thesis

This thesis focuses on Internet Telephony end user services including call processing services and presence services specified in CPL and CPL extensions. For the first time, the basic concepts of presence systems are systematically described; end user's specific services and system basic services are clearly separated. Presence information is extended from only "online" and "offline" indicators to a much broader meaning including "location", "role", "lineStatus" and "availability". CPL is extended to be able to describe presence related services. Through CPL extensions, end users can have their own specific presence services and new call processing services related to presence. We have developed a simulation system to demonstrate Internet Telephony services specified in CPL and its extensions. The simulation system allows end users to specify their services through Graphic User Interfaces (GUIs) and access their services at any locations through the Internet. The thesis contributions include:

1. Describe an architecture and protocols for Presence System (Section 3.1 - 6)

For the first time, presence systems and presence services are described systematically in this thesis. End user services and system basic services are clearly separated in a three-layer architecture. In the architecture, SIP services reside in the lowest layer to support presence system services. The presence services include system basic services as well as end user specific services. Basic services reside in the second layer and are displayed in

various scenarios. End user specific services reside in the third layer, and this leads to the next three contributions.

2. Extend Presence Information (Section 4.3)

The richness of presence information is the basis for presence services. We have extended presence information from traditional "online" and "offline" indicators to a much broader meaning including "location", "role", "lineStatus" and "availability". The whole procedure of the extensions is demonstrated through an example. Our contributions include how to define the extensions of presence information, how to write presence documents in Presence Information Data Format (PIDF) and how to declare the extensions in the presence documents.

3. Extend CPL for Presence (Section 4.4)

In order to describe user specific services related to presence, it is necessary to extend CPL for presence. In the extensions, we have defined four top-level actions, five new operations and a presence-switch. The four top-level actions identify four types of services i.e. the watcher's "outgoing-subscription" request processing, the presentity's "incoming-subscription" request processing, the presentity's "outgoing-notification" response processing and the watcher's "incoming-notification" response processing. The presence-switch provides a set of new services that are processed based on a presentity's presence status. The new services can be either presence services or call processing services.

4. Describe User New Services Related to Presence (Section 4.5 - 6)

With the CPL extensions for presence, end user's presence services and call handling services can be processed based on time, address, language, priority, string (in CPL) and presence status of a presentity (in CPL extensions). These new services have been illustrated through various examples.

5. Implement a Software Simulation System (Section 5.1 - 4)

We have implemented a software simulation system to demonstrate Internet Telephony services. The software allows end users to specify their own services including presence services as well as call processing services through the system's Graphic User Interfaces (GUIs). These services are automatically saved as files in extended CPL by the system. End users can access their services anywhere through the Internet.

6.2 Future Work

Several research topics can be pursued in the future in order to improve, enhance and broaden the research on Internet Telephony services, which may include:

1. Improvement of the Simulation System

In the software that we have implemented (see section 6.1), the presence information contains a maximum of three parameters i.e. the location, the phone line status and the status of willingness to communicate. The software is easily extended to contain more parameters such as "role", i.e. the presentity's working status. Some examples concerning the use of role were given in section 4.3.1.

A user's presence based services, either presence services or call processing services, can only be based on his own presence status in the simulation system. The system can be easily extended to provide these services based on the presence status of any presentity of the user.

2. Further Research on Presence Services

The system we propose is limited concerning the flexibility provided to the watchers. We could envisage extensions where a watcher could express his preferences: stating which

parts of presence information he is interested in, when and how often he likes to receive the presence information, in which language the presence information should be written, etc. These preferences will be written in the watcher's SIP SUBSCRIBE request and sent to his presentity. After the presentity' presence agent (P-PA) receives the request, it will provide the presence services according to the watcher's preferences if these preferences do not conflict with the presentity's policies.

3. Combining Call-handling Services with Other Services

This thesis outlines a new approach to describe services that combine call processing services with presence services through CPL extensions. The approach can be extended to other combined services, such as combining call processing services with Email services, instant Messaging services. By extending CPL for email and for instant messaging, end users can have more kinds of combined services. For an example, an email or an instance message can be automatically sent to a callee if a caller gets no answer from the callee.

4. Broaden Internet Telephony as One Type of Internet Service

Internet Telephony services are provided over the Internet. What are the differences and similarities of Internet Telephony compared to other Internet services? If Internet Telephony services are seen as part of the family of the Internet services, new research topics will arise, such as how to combine these services with others, how to manage these Internet services, how to establish service standards, etc. This will be a very interesting research area.

References

[1] “ITU-T Terms and Definitions Database”. Access June 1, 2003, http://www.itu.int/sancho/index.asp

[2] “SS7 Tutorial”. Access June 1, 2003, http://www.pt.com/tutorials/ss7/

[3] “Intelligent Networks (IN)”. Access June 10, 2003, http://www.iec.org/online/tutorials/in/

[4] Mahhub Hassan, Alfandika Nayandoro and Mohammed Atiquzzaman, “Internet Telephony: Services, Technical Challenges and Products”, IEEE Communications Magazine, vol.38, no.4, pp96-103, April 2000

[5] “Understanding Telecommunications”. Access September 20, 2003, http://www.ericsson.com/support/telecom/part-b/index.shtml

[6] J. Rosenberg, H. Schulzrinne, G. Camarillo etc. “RFC3261 SIP: Session Initiation protocol”. Internet Engineering Task Force, RFC 3261, June 2002. [rfc3261]

[7] J. Lennox, H. Schulzrinne “Programming Internet Telephony Services”. Mar. 18, 1999. Access February 2002, http://www.site.uottawa.ca/~djiang/research/CPL/cpl.html

[8] Jonathan Rosenberg “Programming Internet telephony with CPL and CGI” Bell laboratory, Oct.1999. Access February 10, 2002, http://www.site.uottawa.ca/~djiang/research/CPL/cpl.html

[9] J.Lennox and H.Schulzrinne, “CPL: A language for User Control of Internet telephony Service”. Internet Engineering Task Force, Internet Draft, January 15, 2002. [draft-ietf-iptel-cpl]. Access September 20, 2002, http://www.ietf.org/IP-drafts/draft-ietf-Internet telephony-cpl-06.txt

[10] Extensible Markup Language (XML), Access June 5, 2003, http://www.w3.org/XML/

[11] Ismail Dalgic and Hanlin Fang, "Comparison of H.323 and SIP for IP Telephony Signalling", Proc. Of Photonics East, Boston, Massachusetts, Sep. 1999.

[12] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg "RFC2543 SIP: Session Initiation Protocol". Internet Engineering Task Force, RFC 2543, March 1999. [rfc2543]

[13] J. Rosenberg, H. Schulzrinne, G. Camarillo etc. "RFC3261 SIP: Session Initiation Protocol". Internet Engineering Task Force, RFC 3261, June 2002. [rfc3261]

[14] SIMPLE WG, J. Rosenberg of dynamicsoft, "A Presence Event Package for the Session Initiation Protocol (SIP)", Internet Engineering Task Force, Internet Draft, January 31, 2003, Access May 5, 2003, http://www.ietf.org/internet-drafts/draft-ietf-simple-presence-10.txt

[15] M. Day, J. Rosenberg, and H. Sugano, "A model for presence and instant messaging", Internet Engineering Task Force, RFC 2778, Feb. 2000.

[16] XiaoTao Wu, et al. "CPL Extensions for Presence" Internet Engineering Task Force, Internet Draft, June 1, 2000, Access May 1, 2002, http://www.site.uottawa.ca/~djiang/research/CPL/CPL-Extension/draft-wu-cpl- presence-00.txt

[17] Henning Schulzrinne and Jonathan Rosenberg, "The Session Initiation Protocol : Providing Advanced Telephony Services across the Internet", Bell Labs Technical Journal, Vol. 3 no. 4, pp. 144-160, Oct.-Dec. 1998

[18] J.Lennox and H.Schulzrinne,"Call processing language framework and requirements". Internet Engineering Task Force, RFC 2824, May 2000. [rfc2824]

[19] "W3C Architecture Domain, Naming and Addressing: URIs, URLs, …" Access July 5, 2003 http://www.w3.org/Addressing/#background

[20] A. B. Roach of dynamicsoft, "Session Initiation Protocol (SIP)-Specific Event Notification". Internet Engineering Task Force, RFC 3265, June 2002. [rfc3265]

[21] H. Sugano, S. Fujimoto etc. "Presence Information Data Format (PIDF)", Internet Engineering Task Force, Internet Draft May 2003, Access June 5, 2003 http://www.potaroo.net/ietf/ids/draft-ietf-impp-cpim-pidf-08.txt

[22] J. Peterson, "Common Profile for Instant Messaging (CPIM)" Internet Engineering Task Force, Internet Draft, Feb, 2001. Access May, 2002 http://www.ietf.org/internet-drafts/draft-ietf-impp-im-02.txt

[23] M. Day, S. Aggarwal, G. Mohr and J. Vincent, "Instant messaging / presence protocol requirements". Internet Engineering Task Force, RFC 2779, Feburary 2000. [rfc2779]

[24] Yiqun Xu's thesis, "Detecting Feature Interactions and Feature Inconsistencies in CPL", University of Ottawa, September 2003, Access December 5, 2003, http://lotos.csi.uottawa.ca/ftp/pub/Lotos/Theses/

[25] M. Calder, E. Magill, M. Kolberg, S.Reiff-Marganiec. "Feature Interaction: A Critical Review and Considered Forecast". Computer Networks, Volume 41/1, 2003.

[26] J. Cameron and H. Velthuijsen, "Feature Interactions in Telecommunications Systems". IEEE Communications Magazine, Vol. 31, no. 8, 18-23, August 1993.

[27] J. Lennox, X. Wu and H. Schulzrinne, "CPL: A language for User Control of Internet telephony Service". Internet Engineering Task Force, Internet Draft, August, 2003. [draft-ietf-iptel-cpl]. Access December 20, 2003, http://www.ietf.org/internet-drafts/draft-ietf-iptel-cpl-08.txt

[28] Book "Understanding the Session Initiation protocol" by Alan B. Johnston, Artech House, @2001.

Appendix A:
Formal Definition of “application/pidf+xml” [21]

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="urn:ietf:params:xml:ns:pidf"
     xmlns:tns="urn:ietf:params:xml:ns:pidf"
     xmlns:xs="http://www.w3.org/2001/XMLSchema"
     elementFormDefault="qualified"
     attributeFormDefault="unqualified">

  <!-- This import brings in the XML language attribute xml:lang-->
  <xs:import namespace="http://www.w3.org/XML/1998/namespace"
    schemaLocation="http://www.w3.org/2001/xml.xsd"/>

  <xs:element name="presence" type="tns:presence"/>

  <xs:complexType name="presence">
    <xs:sequence>
      <xs:element name="tuple" type="tns:tuple" minOccurs="0"
         maxOccurs="unbounded"/>
      <xs:element name="note" type="tns:note" minOccurs="0"
         maxOccurs="unbounded"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
         maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="entity" type="xs:anyURI" use="required"/>
  </xs:complexType>

  <xs:complexType name="tuple">
    <xs:sequence>
      <xs:element name="status" type="tns:status"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
         maxOccurs="unbounded"/>
      <xs:element name="contact" type="tns:contact" minOccurs="0"/>
      <xs:element name="note" type="tns:note" minOccurs="0"
         maxOccurs="unbounded"/>
      <xs:element name="timestamp" type="xs:dateTime" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="required"/>
  </xs:complexType>

  <xs:complexType name="status">
    <xs:sequence>
      <xs:element name="basic" type="tns:basic" minOccurs="0"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
         maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="basic">
    <xs:restriction base="xs:string">
      <xs:enumeration value="open"/>
      <xs:enumeration value="closed"/>
    </xs:restriction>
  </xs:simpleType>
```

```
  <xs:complexType name="contact">
  <xs:simpleContent>
      <xs:extension base="xs:anyURI">
        <xs:attribute name="priority" type="tns:qvalue"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="note">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute ref="xml:lang"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:simpleType name="qvalue">
    <xs:restriction base="xs:decimal">
      <xs:pattern value="0(.[0-9]{0,3})?"/>
      <xs:pattern value="1(.0{0,3})?"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- Global Attributes -->
  <xs:attribute name="mustUnderstand" type="xs:boolean" default="0">
    <xs:annotation>
      <xs:documentation>
      This attribute may be used on any element within an optional
      PIDF extension to indicate that the corresponding element must
      be understood by the PIDF processor if the enclosing optional
      element is to be handled.
      </xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:schema>
```

Appendix B: The XML DTD for CPL [9]

```
<?xml version="1.0" encoding="US-ASCII" ?>


<!-- Nodes. -->
   <!-- Switch nodes -->
   <!ENTITY % Switch 'address-switch|string-switch|language-switch|
                      time-switch|priority-switch' >

   <!-- Location nodes -->
   <!ENTITY % Location 'location|lookup|remove-location' >

   <!-- Signalling action nodes -->
   <!ENTITY % SignallingAction 'proxy|redirect|reject' >

   <!-- Other actions -->
   <!ENTITY % OtherAction 'mail|log' >

   <!-- Links to subactions -->
   <!ENTITY % Sub 'sub' >

   <!-- Nodes are one of the above four categories, or a subaction.
        This entity (macro) describes the contents of an output.
        Note that a node can be empty, implying default action. -->
   <!ENTITY % Node     '(%Location;|%Switch;|%SignallingAction;|
                     %OtherAction;|%Sub;)?' >


   <!-- Switches: choices a CPL script can make. -->

   <!-- All switches can have an 'otherwise' output. -->
   <!ELEMENT otherwise ( %Node; ) >

   <!-- All switches can have a 'not-present' output. -->
   <!ELEMENT not-present ( %Node; ) >


<!-- Address-switch makes choices based on addresses. -->
<!ELEMENT address-switch ( address*, (not-present, address*)?,
                           otherwise? ) >
   <!-- <not-present> must appear at most once -->
   <!ATTLIST address-switch
      field         CDATA    #REQUIRED
      subfield      CDATA    #IMPLIED >

<!ELEMENT address ( %Node; ) >
   <!ATTLIST address
      is            CDATA    #IMPLIED
      contains      CDATA    #IMPLIED
      subdomain-of  CDATA    #IMPLIED  >
      <!-- Exactly one of these three attributes must appear -->
```

```
<!-- String-switch makes choices based on strings. -->
<!ELEMENT string-switch ( string*, (not-present, string*)?,
                          otherwise? ) >
   <!-- <not-present> must appear at most once -->
   <!ATTLIST string-switch
      field         CDATA    #REQUIRED   >

<!ELEMENT string ( %Node; ) >
   <!ATTLIST string
      is            CDATA    #IMPLIED
      contains      CDATA    #IMPLIED   >
   <!-- Exactly one of these two attributes must appear -->


<!-- Language-switch makes choices based on the originator's preferred
        languages. -->

<!ELEMENT language-switch ( language*, (not-present, language*)?,
                              otherwise? ) >
<!-- <not-present> must appear at most once -->

<!ELEMENT language ( %Node; ) >
   <!ATTLIST language
      matches      CDATA     #REQUIRED   >


<!-- Time-switch makes choices based on the current time. -->
<!ELEMENT time-switch ( time*, (not-present, time*)?, otherwise? ) >
   <!ATTLIST time-switch
      tzid          CDATA    #IMPLIED
      tzurl         CDATA    #IMPLIED   >


<!ELEMENT time ( %Node; ) >
<!-- Exactly one of the two attributes "dtend" and "duration"
    must occur. -->
<!-- The value of "freq" is (daily|weekly|monthly|yearly).  It is
        case-insensitive, so it is not given as a DTD switch. -->
<!-- None of the attributes following freq are meaningful unless freq
         appears. -->
<!-- The value of "wkst" is (MO|TU|WE|TH|FR|SA|SU).  It is
           case-insensitive, so it is not given as a DTD switch. -->
   <!ATTLIST time
      dtstart       CDATA  #REQUIRED
      dtend         CDATA  #IMPLIED
      duration      CDATA  #IMPLIED
      freq          CDATA  #IMPLIED
      until         CDATA  #IMPLIED
      count         CDATA  #IMPLIED
      interval      CDATA  "1"
      bysecond      CDATA  #IMPLIED
      byminute      CDATA  #IMPLIED
      byhour        CDATA  #IMPLIED
      byday         CDATA  #IMPLIED
      bymonthday    CDATA  #IMPLIED
      byyearday     CDATA  #IMPLIED
```

```
      byweekno      CDATA  #IMPLIED
      bymonth       CDATA  #IMPLIED
      wkst          CDATA  "MO"
      bysetpos      CDATA  #IMPLIED   >


<!-- Priority-switch makes choices based on message priority. -->
<!ELEMENT priority-switch ( priority*, (not-present, priority*)?,
                            otherwise? ) >
<!-- <not-present> must appear at most once -->
<!ENTITY % PriorityVal '(emergency|urgent|normal|non-urgent)' >
<!ELEMENT priority ( %Node; ) >

   <!-- Exactly one of these three attributes must appear -->
   <!ATTLIST priority
      less          %PriorityVal;  #IMPLIED
      greater       %PriorityVal;  #IMPLIED
      equal         CDATA          #IMPLIED   >


<!-- Locations: ways to specify the location a subsequent action
     (proxy, redirect) will attempt to contact. -->

<!ENTITY % Clear  'clear (yes|no) "no"' >


<!ELEMENT location ( %Node; ) >
   <!ATTLIST location
      url           CDATA    #REQUIRED
      priority      CDATA    #IMPLIED
      %Clear;   >
   <!-- priority is in the range  0.0 - 1.0.  Its default value SHOULD
         be 1.0 -->

<!ELEMENT lookup ( success?,notfound?,failure? ) >
   <!ATTLIST lookup
     source         CDATA     #REQUIRED
     timeout        CDATA     "30"
     use            CDATA     #IMPLIED
     ignore         CDATA     #IMPLIED
     %Clear;   >

<!ELEMENT success  ( %Node; ) >
<!ELEMENT notfound ( %Node; ) >
<!ELEMENT failure ( %Node; ) >

<!ELEMENT remove-location ( %Node; ) >
   <!ATTLIST remove-location
      param         CDATA    #IMPLIED
      value         CDATA    #IMPLIED
      location      CDATA    #IMPLIED   >


<!-- Signalling Actions: call-signalling actions the script can
        take. -->

<!ELEMENT proxy ( busy?,noanswer?,redirection?,failure?,default? ) >
```

```
   <!-- The default value of timeout is "20" if the <noanswer> output
        exists. -->
   <!ATTLIST proxy
      timeout       CDATA    #IMPLIED
      recurse       (yes|no) "yes"
      ordering      (parallel|sequential|first-only) "parallel"   >

   <!ELEMENT busy ( %Node; ) >
   <!ELEMENT noanswer ( %Node; ) >
   <!ELEMENT redirection ( %Node; ) >
   <!-- "failure" repeats from lookup, above. -->
   <!ELEMENT default ( %Node; ) >
   <!ELEMENT redirect EMPTY >
   <!ATTLIST redirect
      permanent     (yes|no) "no"   >


   <!-- Statuses we can return -->
   <!ELEMENT reject EMPTY >
   <!-- The value of "status" is (busy|notfound|reject|error), or a SIP
        4xx-6xx status. -->
   <!ATTLIST reject
      status        CDATA    #REQUIRED
      reason        CDATA    #IMPLIED   >

   <!-- Non-signalling actions: actions that don't affect the call -->

   <!ELEMENT mail ( %Node; ) >
   <!ATTLIST mail
      url           CDATA    #REQUIRED   >

   <!ELEMENT log ( %Node; ) >
   <!ATTLIST log
      name          CDATA    #IMPLIED
      comment       CDATA    #IMPLIED   >


<!-- Calls to subactions. -->

<!ELEMENT sub EMPTY >
   <!ATTLIST sub
      ref           IDREF    #REQUIRED   >


<!-- Ancillary data -->
<!ENTITY % Ancillary 'ancillary?' >
<!ELEMENT ancillary EMPTY >

<!-- Subactions -->
<!ENTITY % Subactions 'subaction*' >
<!ELEMENT subaction ( %Node; )>
   <!ATTLIST subaction
      id            ID       #REQUIRED   >


<!-- Top-level actions -->
```

```
<!ENTITY % TopLevelActions 'outgoing?,incoming?' >
   <!ELEMENT outgoing ( %Node; )>
   <!ELEMENT incoming ( %Node; )>

<!-- The top-level element of the script defined for extensions. -->
<!ELEMENT cpl  ( %Ancillary;,%Subactions;,%TopLevelActions; ) >
   <!ATTLIST cpl
             xmlns     %URI;     #REQUIRED>
```