PERSONALIZATION OF INTERNET TELEPHONY SERVICES FOR PRESENCE WITH SIP AND EXTENDED CPL

Dongmei Jiang[1], Ramiro Liscano[2], Luigi Logrippo[1,3]

[1] School of Information Technology and Engineering, University of Ottawa, Canada
[2]Faculty of Engineering and Applied Science, University of Ontario Institute of Technology
[3] Département d'informatique et ingénierie,Université du Québec en Outaouais, Canada
[1] {djiang, rliscano, luigi}@site.uottawa.ca

Abstract— **This paper discusses issues of personalization of presence services in the context of Internet Telephony. Such services take into consideration the willingness and ability of a user to communicate in a network, as well as possibly other factors such as time, address, etc. Via a three-layer service architecture for communications in the Session Initiation Protocol (SIP) standard, presence system basic services and personalized services (personal policies) are clearly separated and discussed. To enrich presence related services, presence information is illustratively extended from the well known "online" and "offline" indicators to a much broader meaning that includes "location", "lineStatus", "role", "availability" etc. Based on this, the Call Processing Language (CPL) is extended in order to describe presence related personalized services for both call processing systems and presence systems using information such as a person's presence status, time, address, language, or any of their combinations. A web-based system is designed and implemented to simulate these advanced services. In the implementation, personal policies are programmed by end users via a Graphic User Interfaces (GUIs) and are automatically translated into extended CPL. The simulation system clearly displays when, where and what CPL policies should be used for the provision of personalized presence services and call processing services. Policy conflicts are also addressed by setting policy priorities in the system.**

Keywords: **SIP; presence; PIDF, presence extensions; CPL; CPL extensions**

1. Introduction

1.1 Presence

Presence in communications conveys the willingness and the ability of a user to communicate with others on a network. "Presence" has been called "the best thing that ever happened to voice" by Jonathan Rosenberg, one of the Session Initiation Protocol (SIP) authors. With awareness of the presence information of other users, unwanted and interrupting calls can be avoided and presence information is very helpful to establish successful communication sessions. The RFC 2778 [1] from Internet Engineering Task and Force (IETF) defines a model and terminology for describing systems that provide presence information. A SIP implementation of this model defines a presence event package for SIP [2]. In this model, a presence system is a presence service that accepts, stores, and delivers presence information to the interested parties defined as *watchers*.

An ETSI/Parlay standard deals with presence services in Parlay X Open Services Architecture (OSA) [3]. The OSA specifications define an architecture that enables application developers to make use of network functionalities through an open standardized interface, i.e. the OSA APIs. [3] specifies in detail a number of presence service scenarios.

Currently, the main advantage of Internet Telephony (or Voice over IP, VoIP) [4] is lower infrastructure costs over conventional telephony systems. Of course another very important characteristic is its ability to handle multimedia communications and presence. Multimedia communications with presence can provide uninterrupted multimedia services via the formats of instant messaging, audio call, video call, multiparty conferencing, etc. After "I Seek You" (ICQ) was introduced in 1996, numerous variations of instant messaging with presence, and more recently, presence based multimedia communications have come to the market very quickly.

1.2 SIP

Currently, there are two main signaling standards in the Internet telephony world: SIP from IETF and H.323 [5] from the International Telecommunications Union -Telecommunications Standard Sector (ITU-T). The Session Initiation Protocol (SIP) [5][6][7] has become a dominant signaling standard because of its simplicity. It is the standard that will be considered in this paper, although our concepts apply to either standard. SIP is an application layer protocol responsible for establishing, modifying and terminating multimedia sessions or calls. Defined on top of a transport layer (TCP or UDP), SIP messages can convey arbitrary payloads: session description, instant message, presence document, JPEG and MIME type. SIP, an end-to-end protocol, can be made available to end user devices to makes it possible to define new personalized services. These services are able to combine conventional 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. One of the main challenges of Internet Telephony is service programming [8]. Built on top of SIP, CPL is a solution created in the IETF for end users to describe and control their specific services.

The advantage of agreeing on standard protocols and languages such as SIP or CPL is the fact that such languages provide commonly understood signals and interfaces by which different implementers can build and exchange applications.

1.3 CPL

CPL [9] was accepted as a proposed standard from the IETF in 2004. It is designed for end users to describe and control their specific telephony services. CPL itself is a very simple programming language in Extensible Markup Language (XML) syntax [10]. CPL does not have variables or loops and can only access limited resources. It is designed to be safe for non-professional users to describe their personalized policies. A CPL script represents a tree of decisions in terms of tags of nodes and links. Each node or link corresponds to a tag in CPL. A node (such as <reject>, <address-switch>) specifies an action to take or a decision to make. A link (such as <address>) specifies the result of an action and displays which decision was taken. CPL is independent of signaling protocols. It can work on top of either the IETF SIP or ITU-T H.323 [5]. As a simple example, the policy for Alice's feature "rejecting anonymous incoming calls" is shown in Fig. 1.

```
<cpl>
  <incoming>
    <address-switch field="origin" subfield="user">
    <!- decision made by checking the original address of the caller -->
    <address is="anonymous">
      <reject/>
<!- if the caller's  name is unavailable, action "reject" is taken and the script
stops -->
    </address>
    </address-switch>
  </incoming>
</cpl>
```

Fig. 1 Screening Anonymous Incoming Call

A CPL script (policy) can be executed on a user's device or on a proxy server that acts on behalf of the user. It is associated with or owned by a particular user, i.e. it is only triggered if the request (e.g. a SIP INVITE) is for that user. In the above example the script would be associated with a particular user (Alice) and the script is executed when a signaling request is received for an incoming call (defined by the <incoming> tag) for that user, Alice. As well, a CPL script can react to a <outgoing> signaling request message. Current CPL can only describe and control call processing services with two directions "incoming" and "outgoing" considered.

1.4 Motivation and Contributions

Currently available presence systems such as Microsoft MSN Messenger or Yahoo Messenger can provide presence information only in one parameter, i.e. an indicator of "online" or "offline" etc., which is too limited to offer rich services related to presence. Personalized services such as how to process a watcher's request and how to notify a watcher based on a person's status, time, address, etc. are not offered to users in current available systems. This paper will overcome these limitations. Our contributions mainly consist of four parts:

1) By proposing a three-layer service architecture, system basic services and personalized services can be clearly separated and described.

2) To enrich presence information and presence related services, we extend presence information and CPL for presence.

3) On the basis of the above extensions, presence related new services can be implemented that take into consideration a person's status, time, language, priority, address or any of their combinations in both a call processing system and a presence system.

4) We have implemented a simulation system to demonstrate these advanced presence related Internet Telephony services written in extended CPL.

While [3] describes a Web service, our proposed presence system is not dependent on a Web implementation. However, the simulation of our presence services is done in a Web environment.

References [11][12] present the language LESS, which is an extension of CPL and is conceived of for end system service creation (while CPL is for service creation on proxies). LESS was developed simultaneously with the CPL extensions presented in this paper. It provides a different approach to the programming of presence services.

Preliminary accounts on this research were presented in [13][14]. For the advanced presence services with SIP, [15] made a comparison between our CPL extensions and the Presence and Availability Policy Language (PAPL) designed at Mitel [24]. Security solutions for the services explained in this paper were proposed in [16].

1.5 Organization

In section 2, we will propose a three-layered architecture in which basic concepts of SIP communication systems (e.g. call processing and presence) are introduced. The system basic services and personalized services are clearly separated and their relationship is addressed in the architecture. The enrichment of presence information, how and in what format the presence information can be carried, are illustrated in section 3. To describe and control presence related new services, CPL will be extended for presence in section 4. In section 5, new advanced services, especially presence related, are illustrated in extended CPL.

With the creation of various new services in extended CPL, service management (creation, description, control and provision) becomes more complicated. Through a simple service management system described in section 6, a simulation system has been implemented in order to experiment with the new proposed service descriptions. The system provides a simulation environment that can be used as a basis for further

research. Service interactions (conflicts) are addressed as well in the simulation system.

2. Three-Layer SIP Communications

In the three-layer architecture seen in Fig. 2, a call processing system (italic) and a presence system are shown to illustrate how the system we propose works. In a presence system, watcher and presentity are two main logical entities; a watcher subscribes to a presentity, and the presentity stores and projects its presence information to the interested party i.e. the watcher.

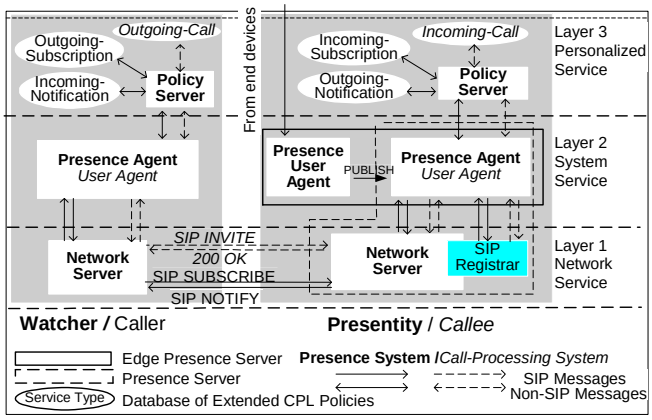

Fig. 2. Presence System and Call processing System Architecture

2.1 SIP System Components

SIP [5] is a client server type protocol modeled after the Simple Mail Transfer Protocol (SMTP) [17] for email and the HyperText Transfer Protocol (HTTP) [18] for the Web. It is responsible for establishing, modifying and terminating multimedia sessions and calls.

SIP has two types of components: user agents (user agent client and user agent server) and network SIP servers (proxy server, redirect server and register server). A user agent acts on behalf of someone who wants to participate in calls or multimedia sessions. A user agent client can initiate a call on behalf of a caller by sending a SIP INVITE [5] message to a callee. A user agent server answers the call on behalf of the callee. A user agent is always addressable with a SIP Uniform Resource Identifier (URI) that uniquely identifies the user. For example, a user agent SIP URI**,** "sip:dongmei@site.uottawa.ca" identifies user "dongmei" at the domain "site.uottawa.ca" over the "sip" protocol. A user agent represents a user to register the user's address to its SIP register server. The address is listed in the "to" header in the SIP REGISTER [5] message. In this way, a SIP server can identify the user agent's current communication addresses. A SIP proxy server or a redirect server directs SIP messages to where they should go for transport. A proxy server acts similarly to a HTTP proxy server when SIP requests are forwarded. A redirect server tells the request client to contact the next hop server directly using a redirect response. A registrar accepts SIP REGISTER requests and keeps the user information to provide location service, which is the key to achieve mobility.

2.2 Concepts of presence system

To offer presence communication in the SIP standard, two SIP extension messages for presence (SIP SUBSCRIBE and SIP NOTIFY) are defined and discussed in [2][19]. The detail of the two messages will be described in Section 2.3. Recently an event state publication method SIP PUBLISH [20] has been proposed in the IETF. This SIP PUBLISH can be used for presence event publication in a presence system. As shown in Fig. 2, a presence system is more complex than a call system. There are two types of user agents in a presence system: presence user agent and presence agent.

A presence user agent is a SIP user agent in a presence system. It takes care of a presentity's end devices (phone, cell phone or Personal Digital Assistant etc.), it manipulates presence information and delivers the presence data provided by the end devices to its presence agent. For example, if a presentity has a phone, the presence user agent takes care of the phone. Upon registration or changes of phone status, the presence user agent publishes the phone status information by a SIP PUBLISH [20] to the presence agent.

A presence agent is a SIP user agent who acts for a watcher or a presentity. It can initiate SIP SUBSCRIBE requests for a watcher to its presentity; for a presentity, it can authenticate and authorize its watchers and supports the presence event package that delivers SIP NOTIFY messages containing presence information in the Presence Information Data Format (PIDF) [21] to its registered watchers.

Notification [2][20] is an unsynchronized process triggered by any process of a watcher successful subscription, a presentity successful registration or any a presentity update of its presence information. A presentity-side presence agent has knowledge of the presence state of its presentity. It can also access presence data manipulated by its presence user agents to generate notifications to all its registered watchers. A presence agent works for one presentity only; however, a presentity can have multiple presence agents with each of them handling some subset of active subscriptions for the presentity. For the case of multiple presence agents, a manager is needed to manage these presence agents. For simplicity, we limit ourselves to the case of a presentity with only one presence agent in this paper.

2.3 Three-Layer Service Architecture

In the three-layer service architecture shown in Fig. 2, SIP servers provide network services in Layer 1. In Layer 2, user agents (or presence agents in the presence system) provide

system basic services to all users fairly. The system services include sending requests for a caller (or a watcher in the presence system), replying to the call for a callee, or authenticating and authorizing the watcher's request in the presence system, and notifying the watcher once its request is approved or its presence status is changed. In Layer 3, personal policies (ovals in Fig.2) are described in extended CPL [22]. These policies are associated with and owned by a particular user and are triggered only when the request is for the user. As a simple example of personal policies, Tom can reject calls from anonymous callers, as shown in the CPL example in figure 1. Personalized policies are programmed by end users, managed by a policy server and executed by user agents or presence agents. Only system basic services will be provided if personal policies are not available.

The policy server works on the third layer to manage personalized services (CPL polices) either for presence or call processing. The management includes creating, storing, updating, deleting, searching, fetching these policies for user presence agents or user agents.

There are two types of call processing policies in current CPL that apply to the SIP INVITE message: incoming-call and outgoing-call policies. Policies for outgoing-call are used to direct callers' user agents when they initiate calls (i.e. send SIP INVITE); and policies for incoming-call are used to direct callees' user agents response when they receive calls (i.e. receive SIP INVITE). Analogous to the call processing system, the presence system has four types of policies: outgoing-subscription, incoming-subscription, outgoing-notification and incoming-notification. These four types of policies are part of our contributions that will be described in section 4 (CPL extensions for presence).

2.4 Scenarios of Presence Processes

SIP SUBSCRIBE and SIP NOTIFY are two operations defined as part of the SIP extensions for presence [2][19]. The SUBSCRIBE operation is a request operation initiated by a watcher's presence agent and its message is routed by SIP network servers to a destination presentity's presence agent. A SIP SUBSCRIBE message contains the required format "application/pidf+xml" of presence information under the header "Accept" and the SIP URI of the destination presentity identified under the header "To". After the presentity's presence agent receives the SIP SUBSCRIBE message, it must authenticate and authorize the subscription request. Once the authentication and authorization are passed successfully, 200 OK is sent back to the watcher presence agent and the watcher is registered successfully to the presentity. The SUBSCRIBE request operation initiates a presence service "dialog" between the watcher presence agent and the presentity presence agent. Notification, an asynchronous request process, is immediately triggered by the successful subscription. The SIP NOTIFY request operation is initialized by a presentity's presence agent and its message is routed by SIP network servers to its watcher agent. The presentity's presence information written in the required PIDF [21] is carried in the body of a SIP NOTIFY [19] message.

We assume that a watcher and a presentity have registered to SIP registrars through their presence agents. The five main process scenarios of presence systems with applicable types of policies are listed following the pattern: scenario description ([scenario actor], [goal], [SIP messages used], [applicable policies]).

1) A watcher sending subscription requests to a presentity ([the watcher's presence agent], [to send a request for the presentity's presence information], [outgoing SIP SUBSCRIBE], [outgoing-subscription]);

2) A presentity processing a watcher's request ([the presentity's presence agent], [to authenticate and authorize the watcher for the request], [incoming SIP SUBSCRIBE], [incoming-subscription]);

3) A presentity notifying a watcher of its presence information ([the presentity's presence agent], [to notify authorized watchers of its presence info], [outgoing SIP NOTIFY], [outgoing-notification]);

4) A watcher receiving a notification ([the watcher's presence agent], [to process arrived notification of presence], [incoming SIP SUBSCRIBE], [incoming-notification]);

5) A presentity resending notifications to its watchers when its presence status changes: this is the same as scenario 3).

3. Presence Information Extensions

Presence information is the basis of presence services. As mentioned in section 1.4, in most systems nowadays presence information is described by only one parameter, an indicator of "on-line" or "off-line", which is very limited to describe a person's communication status. In fact, many parameters can be helpful to indicate whether a user is available to communicate, and different organizations may have different needs." Paper [22] proposed extensions including four new parameters "location", "lineStatus", "role" and "availability", which are basic presence attributes in the communication world. Many other such extensions can be found in [3].

The *location* parameter indicates a presentity's current location. It is easy to understand that a user won't like to be bothered by calls while he is in a meeting in a meeting room; or a user may like to forward his incoming-calls to the room where he is. The parameter value can be "office", "meeting room" or "car" etc. Customers are allowed to define the values of this parameter for their specific purpose. The way of providing location information is flexible. It can be realized automatically with a location sensor system or manually via simple text input etc. In automatic systems, locations must be equipped with sensors in a network. The sensors can identify and monitor the presentity that has registered in the network.

lineStatus indicates if a presentity is occupying a telephone line or not. If we know that a person is talking on a phone, usually we won't interrupt him with other calls. Parameter

"lineStatus" has two possible values, "on" and "off". Value "on" indicates that the user is on a phone and value "off" indicates that the user is not on a phone.

role indicates the role of a presentity in an organization, such as "professor", "consultant", etc. A user can have multiple roles when the user takes more than one positions. For example, Sharon works as a consultant for a company one day a week and works as a professor for a university four days a week. She has two roles, "consultant" and "professor". Sharon needs a service that all calls from her company are be blocked unless she takes the role of "consultant". The role status is an important factor to affect a user's ability and willingness to communicate.

Availability indicates whether a presentity wishes to communicate with others. The value "yes" indicates that the user is ready to communicate with others and "no" indicates that the user does not wish to communicate with others currently. The availability value "no" is analogous to the notice tag "Please do not bother" that can be tied to the room door handle in hotels. It is only a sign to show if a user is available to communicate. When a user has a status of "yes", other users have a better chance to successfully communicate with her than if she has a status of "no". Availability is not the presentity's communicating ability; it is his willingness. On status "no", the user can still communicate with others if necessary. For example, he can still answer emergency phone calls.

Based on the above extensions, presence status can be described with multiple parameters. When any of the presence parameters is changed, the user's presence status is changed and his presence agent is informed of this change. The presence agent then notifies all watchers of his updated presence information according to the user's notification policies.

```
<?xml version="1.0" encoding="UTF-8"?>
 <xs:schema ......>

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

  <! definition of "lineStatus"  -->
  <xs:simpleType name="lineStatus">
   <xs:restriction base="xs:string">
    <xs:enumeration value="on"/>
    <xs:enumeration value="off"/>
   </xs:restriction>
  </xs:simpleType>

  <! definition of "role"  -->
  <xs:simpleType name="role">
   <xs:restriction base="xs:string">
    <xs:enumeration value="student"/>
    <xs:enumeration value="consultant"/>
    <xs:enumeration value="professor"/>
   </xs:restriction>
  </xs:simpleType>

  <! definition of "availability" -->
  <xs:simpleType name="availability">
   <xs:restriction base="xs:string">
    <xs:enumeration value="yes"/>
    <xs:enumeration value="no"/>
   </xs:restriction>
  </xs:simpleType>
 </xs:schema>
```

Fig. 3. Example of Definition of Presence Extensions

Fig. 3 shows the definition of the above extensions of presence in the format of a XML schema. In the same way, organizations may define their own specific presence extensions.

Presence information is written in presence documents in PIDF [21] using XML syntax. The documents are carried in the body of SIP NOTIFY messages sent to watchers. As an example of presence extensions, Fig. 4 shows Bob's presence information sent to Alice, wrapped in a SIP NOTIFY message. The presence document indicates that Bob can accept instant messages; he is working as a professor in his office and he is talking on his phone; also he prefers not to communicate with others at this time (he is not available). Knowing this, Alice decides not to bother Bob with a call.

```
NOTIFY sip ……

<presence …>
 <tuple id="…">

  <status>
   <basic>open</basic>
    <!- able accept instant message? -->
   <epidf:location> office </epidf:location>
   <epidf:lineStatus> on </epidf:lineStatus>
    <!- talking on a phone? -->
   <epidf:role> professor </epidf:role>
   <epidf:availability> no </epidf:availability>
    <!- like a talk now? -->
   </status>

  </tuple>
</presence>
```

Fig. 4. Presence in PIDF carried in a SIP NOTIFY

4. CPL Extensions for Presence

Current CPL [9] deals with call processing services, triggered by SIP INVITE messages. The first proposal for extensions of CPL for presence was [23]. This proposal added to CPL the capability to describe presence services, but the focus was on basic system services rather than user personalized services. We have defined a new switch (the presence-switch) in our extensions, which makes it possible to personalize user services in both presence systems and call processing systems. Our CPL extensions make it possible to define presence services with consideration of a person's presence status, i.e. a person's location, line status, role, availability etc. Our work was influenced by previous work at Mitel, leading to a patent application [24], where the use of a CPL-based language for presence policies was described. This latter work produced a language for presence that was inspired by CPL but did not support the existing CPL tags for managing these services. In contrast, our CPL extensions are compatible with current CPL, which makes them much more powerful. With the new presence-switch characterized by rich presence information, end user services are much enhanced. This is very important for service personalization. In either a presence system or a call processing system, user services can be handled with consideration of a person's status (i.e. presence-switch), time (i.e. time-switch), address (i.e. address-switch) etc. or any of their combinations.

In our CPL extensions, we define four top-level actions, five operations and a presence-switch [22].

4.1 Four New CPL Top-level Actions

Just as current CPL is designed to process SIP INVITE, our CPL extensions for presence will be able to process SIP extensions for presence methods, which are SIP SUBSCRIBE and SIP NOTIFY with the two directions of "incoming" and "outgoing". Therefore, to keep consistent with the definitions of the top-level actions "incoming-call" and "outgoing-call" in CPL, the extensions of CPL for presence should define four top-level actions: "*incoming-subscription*", "*outgoing-subscription*", "*incoming-notification*" and "*outgoing-notification*". The four top-level actions, triggered by either a SIP SUBSCRIBE message or a SIP NOTIFY message, induce four types of personalized presence services (policies) shown on the third layer of Fig. 2.

1) incoming-subscription - the action that is performed on the presentity-side when a SIP SUBSCRIBE message arrives and the message's destination is the script owner, i.e. the presentity.

2) outgoing-subscription - the action that is performed on the watcher-side when a SIP SUBSCRIBE message is ready to be sent and the message's originator is the script owner, i.e. the watcher.

3) incoming-notification - the action that is performed on the watcher-side when a SIP NOTIFY message arrives and the message's destination is the script owner, i.e. the watcher.

4) outgoing-notification - the action that is performed on the presentity-side when a SIP NOTIFY message is ready to be sent and the message's originator is the script owner, i.e. the presentity.

4.2 Five New CPL Operations

1) subscribe - this action causes the SIP server to send a SIP SUBSCRIBE message to the specified presentity.

2) notify - this action causes a SIP server to send a SIP NOTIFY message to the specified watcher. The NOTIFY message may contain a presence document in PIDF.

3) approve - this action tells a PA of a presentity that the watcher request is approved with a time limit. The PA then can start to prepare the notification message, the beginning of a notification process.

4) accept – by this action, the PA of a watcher accepts the received notification. The presence information is displayed or refreshed.

5) call – this action combines action "accept" for presence with a new "call" action for the call processing service. It causes the SIP server to send a SIP INVITE message to a specified callee. The script owner is the caller who launches a new call. The action enables the co-operation of the presence system and the call processing system i.e. the presence agent in the presence system co-operates with the SIP user agent in the call processing system.

4.3 A New CPL Presence Switch

1) presence-switch – this switch enables an end user to make decisions based on the presence status of a presentity. By the definition of presence extensions in section 3, presence status can be a user's location, lineStatus, role status availability status or any of their combinations. The presentity may be the user himself or somebody else. This is an important feature. If the presentity is the user himself then the presence information can be acquired from the user's presence user

agent. Otherwise if the presentity is somebody else then the user must have captured the presentity's presence information from a previous SIP NOTIFY message or from the incoming SIP NOTIFY message depending on which event triggers the CPL script.

Node "presence-switch" has two mandatory parameters, "presentity" and "timeout". Parameter "presentity" identifies a presentity, and parameter "timeout" gives the CPL executor (i.e. a user agent in a call processing system or a presence agent in a presence system) a time limit to retrieve presence information. "presence-switch" is followed by the link node "presence", which specifies a presence status to verify whether the presentity matches the status or not.

The definition of CPL extensions for presence can be in the format of either DTD or XML schemas [22]. The scripts in extended CPL will be validated using the definition before they are executed. With these extensions of CPL, personalized services are much enriched in both call processing and presence systems. These services can be handled based on a person's location, lineStatus, role, availability, time, address or any of their combinations.

5. Applications

The new services in extended CPL described in Section 4 are event-driven services. They can be caused by the events of "incoming-call" or "outgoing-call" in call processing systems, or by the events of "incoming-subscription", "outgoing-subscription", "incoming-notification" or "outgoing-notification" in presence systems. We can classify these new services into three main types according to the actions that are finally taken. These types are screening service, forwarding service and auto-call service. The first two types apply to both call processing systems and presence systems. The auto-call service is based on a presence event, is initiated in a presence system and ends up in a call processing system. These types of services are illustrated with the following four examples described in extended CPL.

```
<cpl xmlns = …>
<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>
```

Fig. 5. Conditional Authorization for Presence

The first example is Sharon's policy of conditional authorization to her watchers as shown in Fig. 5. Sharon screens her incoming-subscription requests. She accepts her boss's requests only during working hours, i.e. from 9:00am to 5:00pm, Monday to Friday. This is a "pure" screening example for presence subscription where the trigger is an "incoming" SIP SUBSCRIBE request.

The second example shown in Fig. 6 represents an outgoing-call screening service based on the presence status of Sharon's callee (i.e. her boss). This particular CPL code exemplifies the concept of extending typical CPL top-level actions (<outgoing>, <incoming>) with presence switches (shown in italic). Sharon thinks that she should not call her boss when he is talking on his phone with his availability "no". Sharon makes her outgoing-call screening policy to prevent such calls. In her policy, shown in Fig. 6, Sharon blocks her calls to her boss when the boss is talking on his phone and his availability status is "no". The calls will be processed based on the callee's status, which is triggered by Sharon's "outgoing" SIP INVITE messages.

```
<cpl xmlns ......>
<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>
```

Fig. 6 Screening Outgoing-calls

```
<cpl xmlns ......>
<incoming>

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

</incoming>
</cpl>
```

Fig. 7 Presence-based Incoming-call Forwarding

The third example shown in Fig. 7 is a call-forwarding service for Sharon's boss. He forwards his incoming-calls to his voice mail when he is talking on his phone with his availability status "no". He prefers to deal with these voice

mails at a later time. His policy is shown in Fig. 7. These calls are processed based on callee's (the boss's) status, which is triggered by his "incoming" SIP INVITE messages.

The last example is an auto-call policy of Peter. As shown in Fig. 8, Peter asks to initiate an auto call to Sharon as soon as he is notified that Sharon arrives at her office. This auto-call is based on the callee's (Sharon's) presence status, which is carried in the "incoming" SIP NOTIFY from Sharon to Peter.

```
<cpl xmlns = …> ……
<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>
```

Fig. 8 Presence-based Auto Calls

6. System Design and Implementation

By using the Java programming language, we have designed and implemented a web-based system to simulate personalized call processing services and presence services written in extended CPL. Its main outline is shown in Fig. 9. It contains three subsystems: the presence system, the policy system, and the call processing system. This figure shows a web page where the name of each subsystem is a link leading users to the web page of the corresponding subsystem.

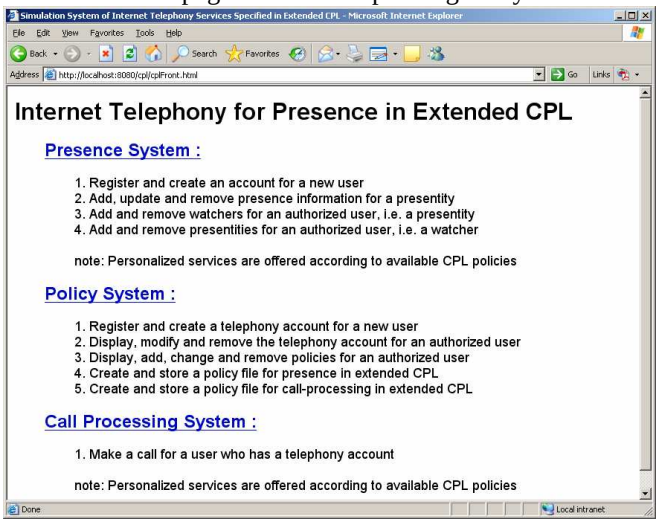

Fig. 9 Simulation System

6.1 System Introduction

The three subsystems can either be independent or cooperate. The call processing system allows a registered user to make phone calls to other registered users. The presence system can deliver presence information to watchers; register, modify or remove presence information for a presentity; add or remove watchers for a presentity; add or remove presentities for a watcher. These are system basic services that treat all users fairly. The policy system can create and modify user telephone accounts; create, modify and remove user policies; translate user policies into extended CPL and store the CPL files in the system. Personalized services in either a presence system or a call processing system will be offered according to the respective CPL policies. System basic services will be provided only if there are no applicable personal CPL policies.

A user needs to create an account in order to login to these subsystems. A user is identified by a unique name with a confirmed password. The user is allowed to have a logical phone as a global user identifier and three end devices: a regular phone, a cell phone and a voice mail, each of which is identified by an address consistent with SIP. These addresses are valuable elements that will be extracted and written into the user's CPL policies.

6.2 System Architecture

The system architecture, shown in Fig. 10, 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 managing the simulation services. The Database Server holds the database agent who works as a representative for the database and has various methods to operate on the database. The database contains all service-related information. User requests are entered via the browser's Graphic User Interfaces (GUIs). The Java servlets accept a request through HTTP and initiate a corresponding request to the database agent that is connected to the database via the Java Database Connection (JDBC). The database agent obtains the required data from the database and sends it back to the Java servlets. The Java servlets process the data and send the readable service results to the user browser. Held by their own servers at different locations as shown in Fig. 10, the database agent and the Java servlets are independent of each other in this architecture, which makes them easy to extend and modify individually.

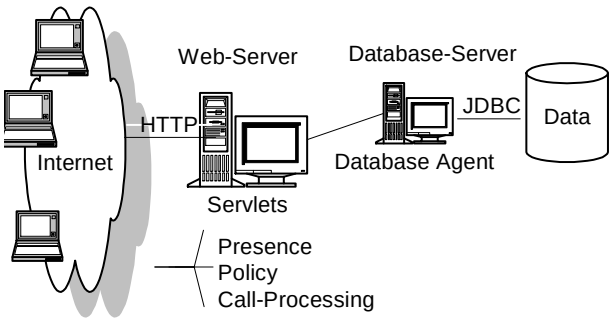

Fig. 10 Simulation System Architecture

6.3 Policy Management

Our simulator system supports six types of policies. With "incoming" and "outgoing" directions considered, the first two types acting on SIP INVITE can be described in current CPL. They are incoming call (IN) and outgoing call (OUT) in a call processing system. The last four types acting on SIP SUBSCRIBE and NOTIFY are described in our CPL extensions. They are incoming subscription request (SIN), outgoing subscription request (SOUT), incoming notification response (NIN) and outgoing notification response (NOUT) in a presence system.

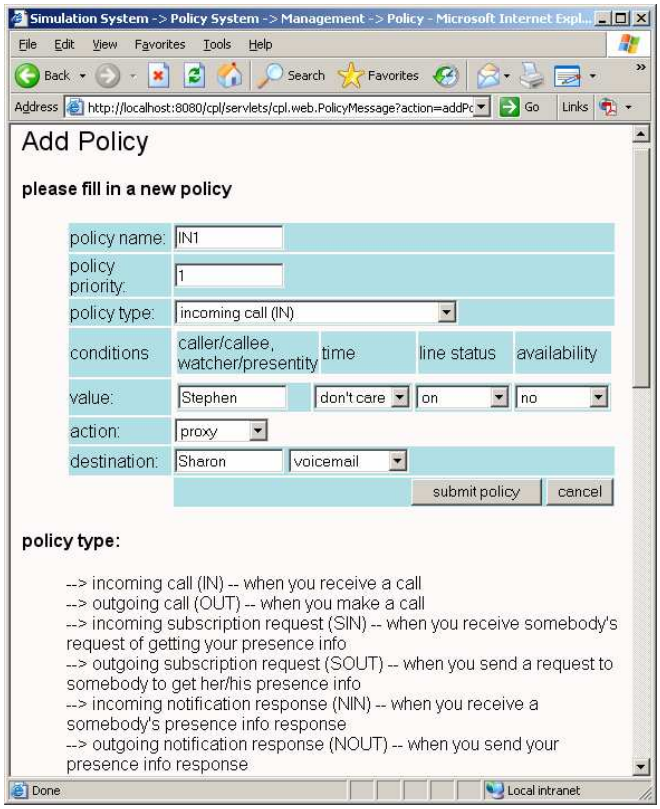

Fig. 11 Policy Creation

A CPL policy in the simulation is composed of the following four parts: type (CPL top action), conditions (CPL switches), one action (CPL action) and destination (CPL location). This is clearly reflected on the policy creation GUI, shown in Fig. 11. The policy name is defined by the end user and the policy priority is set based on the priorities of the different policies within a service type. The policy will be triggered whenever all the conditions are satisfied. The policy in Fig. 11 indicates that all incoming calls from Stephen will be forwarded to Sharon only if she is the policy owner, she is not available and she is on her phone. A more elaborate GUI would allow users to specify service type and their personalized needs in terms of a person's status, time, address or any of their combinations.

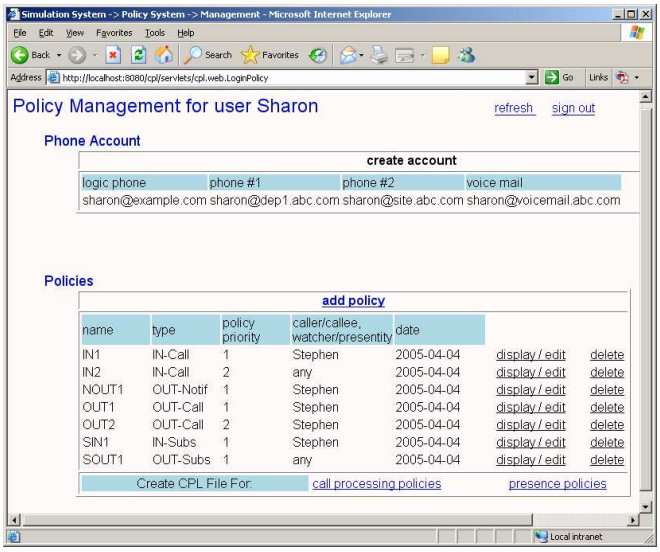

Fig. 12 Policy Management

The policy management GUI will pop up after a user submits his policy. Fig. 12 is the management GUI for user Sharon. By clicking the account related links “edit” and “delete”, Sharon can edit her personal account and deregister her personal information from the system respectively. Using policy related links “add policy”, “edit”, “display” and “delete”, Sharon can add, edit, display and remove her policies. Using the links for policy creation (“call processing policies” and “presence policies”), Sharon’s policies are automatically translated into extended CPL files and stored in the simulation system. These CPL policies will be executable for both Sharon’s call processing services and her presence services.

```
<cpl ..... >
<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>

<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>

<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>
```

Fig. 13 Auto-created Presence Policies in extended CPL

The CPL files of the presence policies and the call processing policies are separately created and stored and they will be executed individually in two different subsystems. Fig. 13 is the CPL file for Sharon’s presence policies automatically created by the simulation system. For example, policy “NOUT1” indicates that Sharon blocks notifications to Stephen if she is on the phone.

6.4 Policy Conflicts

One user can have multiple policies for each policy type. In Fig. 12, Sharon has two incoming call policies and two outgoing call policies. Sometimes there are policy conflicts. For example, Bob prefers to forward his incoming calls to his voice mail from 9:00am to 10:00am, however, he wants to take his wife’s calls unconditionally. A conflict occurs if Bob’s wife calls him at 9:30am. Giving a higher priority to the policy for his wife’s calls, the conflict is solved and Bob is able to take his wife’s calls even at 9:30am In order to eliminate policy conflicts, policies are arranged by numbers when created as shown in Fig. 11: the smaller the number, the higher the policy priority. The policy priority order inside each service type is clearly displayed in the policy manage-

ment GUI as shown in Fig. 12. In the processing of the policies for each service, the highest priority policy is checked first. As soon as a policy matches the criteria, this policy is executed and the remaining policies are ignored.

The study of such service conflicts (called feature interactions in the telephony world) is beyond the scope of this paper. However clearly they will have to be taken into careful consideration by the system designers. In many cases, the naive user unfortunately will not be able to distinguish interactions from system malfunctions. Reviews of research on service conflicts in traditional telecommunications systems can be found in [25][26] and policy conflicts in CPL are discussed in [27][28].

6.5 Presence Management

In the presence system, a user (e.g. Sharon shown in Fig. 14) can be a watcher, a presentity or both at the same time. As a watcher, Sharon can send requests for her presentities' presence information; as a presentity, Sharon can authorize her watchers and notify them of her presence information.

The presence management GUI, shown in Fig. 14, will be popped up when the user (Sharon) logins to the presence system. A user's presence status is characterized by the parameters "location", "lineStatus", "role" and "availability". The presence information, stored in the system database, can be managed by Sharon herself and can be obtained by her authorized watchers according to CPL policies.

Each link in Fig. 14 connects to a presence service on the GUI. For example, by clicking the link "add watcher", the GUI "Add Watcher" pops up allowing the current user Sharon to approve an incoming subscription request from a specified watcher. By clicking the link "add presentity", the "Add Presentity" GUI is popped up allowing Sharon to send out a subscription request to the specified presentity. Sharon can change her presence status via the GUI "Presence Update", which is popped up by clicking the link "edit". She may deregister her presence information from the system by clicking the link "delete". Sharon is allowed to terminate her presence service to her watcher (e.g. Dongmei) by clicking the link "delete" in the row of watcher Dongmei. Sharon also can stop being informed of the status of her presentity (e.g. Christopher) by clicking the responding link "delete".

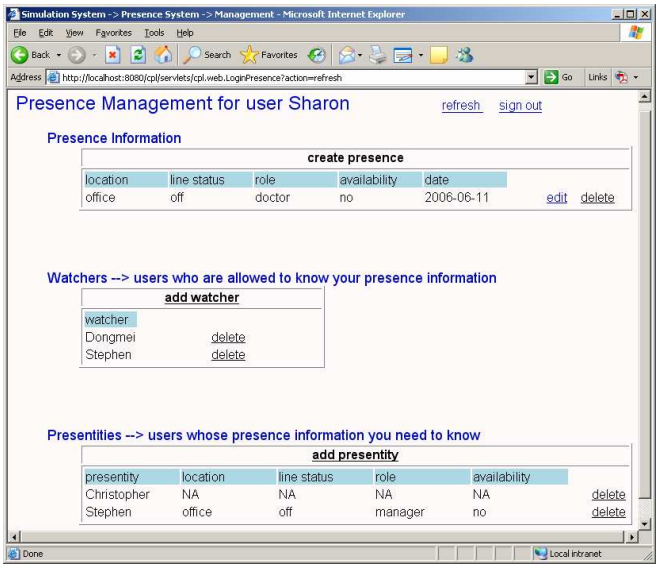

Fig. 14 Presence Management

The presence system will provide user specific services (policies) once a user has his personal CPL policies. In the example of Sharon, she has a conditional outgoing notification policy "NOUT1", shown in Fig. 12, to block her notifications to Stephen when she is talking on her phone during working hours.

A CPL policy is triggered by a service event in the simulation. When Sharon updates her phone line status from "off" to "on" in working hours via the GUI "Presence Update", the result for her presence update event is shown in Fig. 15. Stephen is not notified according to Sharon's policy "NOUT1". The other watcher (Dongmei), for whom Sharon has no notification policy, is successfully notified in the system default behavior.

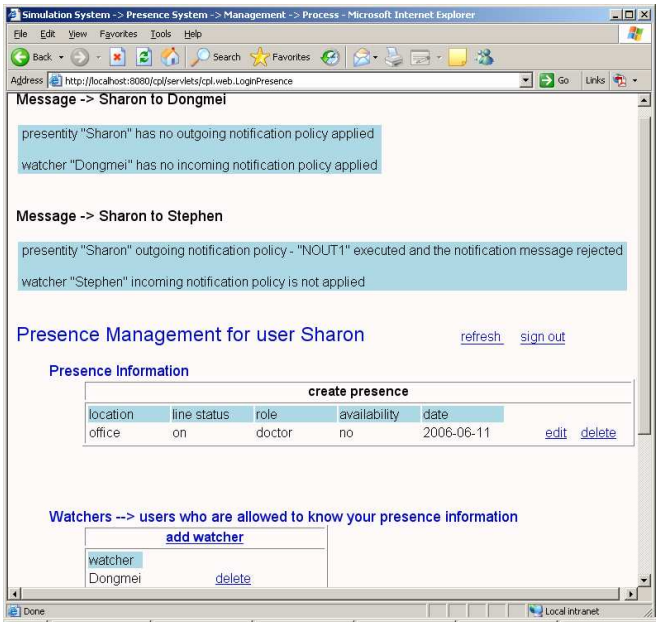

Fig. 15 Policy "NOUT1" Provision

The above scenario clearly illustrates what CPL policies should be used for the provision of presence services. In the call processing system, the personal CPL policies are provided in a similar fashion. If no personal policies are available, system basic services are provided by default.

7. Conclusions

This paper started by proposing an architecture for personalized services for SIP multimedia communications with presence. Some presence information extensions were introduced and on their basis CPL has been extended to deal with these services. With these CPL extensions, end users will have more flexible and powerful call processing services and presence services, which can take into consideration a person's location, phone line status, role status, availability status, time, address, or any combinations of these.

A web-based application system has been designed and implemented to simulate SIP communications with personalized services defined in extended CPL. Via user-friendly GUIs, end users can program their specific services, which are translated into extended CPL automatically by the system. Presence services and call processing services are then simulated according to the user's CPL policies if available. Policy conflicts in one user's set of policies have been solved by a priority mechanism, although the more general topic of policy interaction remains to be studied. This simulation environment can be used as a basis for further research.

ACKNOWLEDGMENT

This work was motivated by a grant from Mitel Networks. It was also funded partially by Communications and Information Technology Ontario and the Natural Sciences and Engineering Research Council of Canada. We thank Tom Gray and Romelia Plesa for having participated in parts of this work and Jacques Sincennes for technical advice.

REFERENCES

[1] M. Day, J. Rosenberg, H. Sugano. A Model for Presence and Instant Messaging, IETF RFC 2778, 2000.

[2] J.Rosenberg. A Presence Event Package for the Session Initiation Protocol (SIP), IETF RFC 3856, 2004.

[3] ETSI. Final draft ETSI ES 202 391-14 V1.1.1 (2005-01), 2005. At URL: http://webapp.etsi.org/action/MV/MV20050318/es_20239114v010101m.pdf. Accessed Jan. 2006.

[4] M. Hassan, A. Nayandoro, M. Atiquzzaman Internet Telephony: Services, Technical Challenges and Products, IEEE Communications Magazine, vol.38, (4), 2000., 96-103.

[5] I. Dalgic, H. Fang. Comparison of H.323 and SIP for IP Telephony Signalling, Proc. Of Photonics East, Boston, Massachusetts, 1999.

[6] J. Rosenberg, H. Schulzrinne, G. Camarillo et al. SIP: Session Initiation protocol, IETF RFC 3261, 2002.

[7] A.B. Johnston. Understanding the Session Initiation protocol, Artech House, 2001.

[8] J. Rosenberg, J. Lennox, H. Schulzrinne Programming Internet Telephony Services, IEEE Internet Computing 3(3), 1999, 63-72.

[9] J. Lennox, X. Wu, H. Schulzrinne. Call Processing Language (CPL): A language for User Control of Internet Telephony Services, IETF RFC3880, 2004.

[10] W3C. Extensible Markup Language (XML), 2003. At URL: http://www.w3.org/XML/. Accessed Jun. 2003.

[11] X. Wu, H. Schulzrinne, LESS: Language for End System Services in Internet Telephon5 Internet Draft-wu-iptel-less, 2005. At URL: http://www.softarmor.com/wgdb/docs/draft-wu-iptel-less-00.txt. Accessed Jan. 2006.

[12] X. Wu, H. Schulzrinne. Handling Feature Interactions in the Language for End Systems Services. To appear in Computer Networks, 2006.

[13] D. Jiang, T. H. Yeap, L. Logrippo, R. Liscano Personalization for SIP Multimedia Communications with Presence, ICSSSM'05, Vol. 2, 1365-1368, 2005,.

[14] D. Jiang, T. H. Yeap, L. Logrippo, R. Liscano. Simulation of Personalized Services in SIP Communications, IEEE ICSSSM'05, Vol. 2, 1379-1382, 2005.

[15] D. Jiang, T. H. Yeap, R. Liscano, L. Logrippo. Two Approaches for Advanced Presence Services in SIP Communications, 2005 IEEE Malaysia International Conference on Communications and IEEE International Conference on Networks (MICC & ICON 2005).

[16] D. Lou, D. Jiang, T. H. Yeap, W. O'Brian. Personalized Service Mobility and Security in SIP-based Communications, 2005 IEEE Malaysia International Conference on Communications and IEEE International Conference on Networks (MICC & ICON 2005).

[17] J. Klensin (Ed.) Simple Mail Transfer Protocol, IETF RFC 2821, 2001.

[18] R. Fielding, J.Mongul, H. Nielsen, T. Berners-Lee. Hypertext Transfer Protocol HTTP/1.1, IETF RFC 2068, 1997.

[19] A.B. Roach. Session Initiation Protocol (SIP)-Specific Event Notification, IETF RFC3265, 2002.

[20] A. Niemi, Ed.. Session Initiation Protocol (SIP) Extension for Event State Publication, IETF RFC 3903, 2004.

[21] H. Sugano, S. Fujimoto et al. Presence Information Data Format (PIDF), IETF RFC 3863, 2004.

[22] D. Jiang. Internet Telephony Services for Presence with SIP and Extended CPL, Master's Thesis, University of Ottawa, 2004.

[23] X.T. Wu et al. CPL Extensions for Presence, IETF Internet Draft., 2000 http://www1.cs.columbia.edu/~xiaotaow/rer/Research/Paper/draft-wu -cpl-presence-00.txt. Accessed, May 2004.

[24] R. Liscano, K. Baker, N. Balaba, J. Zhao. Role-based Presence, UK Patent Submission File #0218711.0, 2002.

[25] S. Reiff-Marganiec, K. Turner. Feature Interactions in Policies. Computer Networks, 45 (2004), 569-584.

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

[27] Y. Xu, L. Logrippo, J. Sincennes, Detecting Feature Interactions in CPL. To appear in the Journal of Network and Computer Applications.

[28] M. Nakamura, P. Leelaprute, K. Matsumoto, T. Kikuno. On Detecting Feature Interactions in Programmable Service Environment of Internet Telephony. Computer Networks, 45(5) (2004) 605-624.