Model driven context awareness

SERGI LAENCINA VERDAGUER

KTH Information and
Communication Technology

Master of Science Thesis
Stockholm, Sweden 2007

COS/CCS 2007-02

Model driven context awareness

Sergi Laencina Verdaguer

2007.01.28

Master of Science thesis performed at
Center for Wireless Systems, KTH
Stockholm, Sweden

Advisor and examiner: Prof. Gerald Q. Maguire Jr.

Abstract

The very nature of mobile phones makes them ideal vehicles to study both individuals and organizations: people habitually carry a mobile phone with them and use it as a medium for much of their communication. The information available from today's phones includes the user's location, people nearby, and communication (call and SMS logs), as well as application usage and phone status (idle, charging, and so on).

The main goal of this project is to combine some of the new technologies of voice over IP (VoIP) with context awareness services for mobile users and create a demonstrator for a typical routine of a student in Kista.

We used context awareness together with the SIP Express Router to make a system more intelligent for the user. In this thesis the definition of CPL scripts and how they could exploit context information to provide SIP service that would be useful to a student were examined. A simple test was conducted to measure the overhead of using context awareness by the SIP proxy when processing CPL scripts.

Sammanfattning

Mobila telefoner gör dem ideala medel för att studera både individer och organisationar: personer bär ofta en mobil telefon med dem och använder den som ett medel för mycket av deras kommunikation. Informationen som är tillgänglig från dagens telefoner inkluderar användares läge, personer som är närliggande och kommunikation, såväl som applikationanvändning och telefon status.

Målet av detta projekt är att kombinera som några av de nya teknologierna av röst över IP (VoIP) med kontextuppmärksamma servar för mobila användare och skapar en demonstrant för en typisk rutin av en studerande i Kista.

Vi använde kontextuppmärksamma med SIP Express Router för att göra ett system mer intelligent för användare. I detta examensarbetet undersöker vi CPL skrifter och hur de skulle kunna exploatera kontext information för att ge den SIP tjänsten som är användbar till en studerande. Ett enkelt test förades för att mäta det över huvudet av att använda kontextuppmärksamma av den SIP proxyen när det arbetar med CPL skrifter.

Acknowledgements

First of all I would like to express my most sincere gratitude to my project advisor, Professor Gerald Q. Maguire Jr., for helping me when I needed, encouraging me when problems rose, answering all my doubts, and being always willing to transmit his positivism and share his knowledge. Secondly, I would like to thank to Alisa Devlic, because she has been always willing to give a hand in the worse moments, and has led my project into a successful ending.

I would like also to thank all my colleagues at the lab for maintaining such a good atmosphere that made work easier and more comfortable, and specially my friend Oscar.

All my friends need to be mentioned here, those who were encouraging me from Spain and those who were here, especially all my friends from Vasagatan 9.

My family, who have always been there, in the good, the bad, and the worse moments, and had always believed in me. It is for that reason that I would like to thank deeply my father Benjamí, my mother Rosa Maria and my sister Eulàlia.

And to thank everybody that has contributed to this project, sharing their knowledge and devoting some of their time to help me carry out this challenging task.

Table of Contents

List of Figures and tables

Acronyms

AP	Access Point
CPL	Call Processing Language
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DTD	Document Type Definition
GPRS	General Packet Radio Services
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
OWL	Web Ontology Language
PBX	Private Branch eXchange
PDA	Personal Digital Assistant
RADIUS	Remote Authentication Dial In User Service
RDF	Resource Description Framework
SCTP	Stream Control Transmission Protocol
SER	SIP Express Router
SGML	Standard Generalized Markup Language
SIP	Session Initiation Protocol
SIPS	Session Initiation Protocol Security
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UA	User Agent
UAC	User Agent Client
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
VoIP	Voice over Internet Protocol
XML	Extensible Markup Language

1. Introduction

Computer capabilities and network technology have been increasing for many decades. Today, computers offer a lot of functionality even in tiny handheld devices. Moreover, the increasingly ubiquitous wireless communications offers users of these devices to be always online and able to access information at any time and anywhere. In addition, the bandwidth of wireless links keeps increasing, providing an increasingly large range of potential services.

Context-awareness is part of this new intelligent network that allows the system to act based on the user's current environment and situation (his context). This means that when humans interact with other persons or in specific situations in a given environment, we make use of implicit situational information. If the current situation context can be deduced and interpreted appropriately, then the system can automatically adapt to "do the right thing" for the user. For example, if a person wants to go to buy something from a shop, but it is raining when he leaves his house. It is likely that, this person will return to his house and take an umbrella. This occurs because the person is aware of his environment, situation, and context and reacts appropriately.

Unfortunately, computers are not as good as humans in deducing situational information from their environment and in using it to facilitate their interactions with users and others computers. Thus they cannot easily take advantage of such information in a transparent way, but if they can do so, it usually requires that this context information be explicitly provided. This is a challenge for computer systems as well as a barrier to better human-computer interaction.

Context-aware computing tries to provide this additional context information to our applications, by discovering and taking advantage of contextual information such as user location, time of day, nearby people and devices, etc.

But before going further, it is important to understand the concept of context and context awareness.

1.1 What is context?

First of all we need to define the term "context". Many definitions of "context" can be found in the literature and you may get a new definition from each person you ask. The first definition given in the Compact Oxford English Dictionary says:

"context
• **noun 1** the circumstances that form the setting for an event, statement, or idea." [1]

This is closely related to the definition that Dey and Abowd use:

"Context is any information that can be used to characterize the situation of an entity, where an entity can be a person, place, physical or computational object". [2]

In fact, we can define several types of context to process with computers. Göker and Myrhaug [3] proposed the following categories:

- Location context:	Where is the user? (At work, at home …). What time is it? (19:00, lunch time,…). Location is one of the most useful and widely used types of context information in context aware applications.
- Environment context:	Information related to the user's surroundings and environment in a very general sense, such as the available networks or the ambient temperature.
- Social context:	The user's friends and relatives, his contact list, the user's social position…
- User preferences:	These are often specified by a user or service profile (i.e. what the user likes and dislikes, the languages the user speaks,…)
- Personal context:	What is the user interested in? (Hobbies, sports,…). Is the user diabetic, does the user have allergies?
- Task context:	What is the user doing or what will the user be doing. What had the user planned to do?

Now that we understand the meaning of context we can explain concept of a context aware service.

1.2 What is context aware service?

Context awareness is based on using context information in order to make decisions. Thus we can define context aware service as a service that uses context information as an input and adjusts it behaviour accordingly. Examples of such services are:

- Using a presence detector, if the room is empty, then the temperature can be automatically regulated and the lights turned off to save energy.

- If a PDA can support both WLAN and GPRS connectivity, then which interface is used can be automatically changed according to the user's preferences and locations: for example, when a user is in a WLAN coverage, his PDA should switch to the WLAN network instead of using GPRS. Additionally he should use his VoIP application rather than make a circuit-switched GSM call.

- If the user is giving a talk at a conference, he wants that all incoming calls to be redirected to his voice mail automatically unless it is an emergency call.

Which applications can actually be implemented? Which applications are “low hanging fruit” and could easily be implemented and offer high return on the investment?

1.3 Context Aware Applications

While there are plenty of context-aware applications, most of them have not been yet commercialized. Some of the existing applications include office and meeting tools, tourist guides, context-aware fieldwork tools, memory aids and a framework for developing context-aware applications [4].

An obvious context-aware application is a location-based restaurant guide. Such a location-based restaurant guide is useful for obtaining a list of the restaurants near the user’s current location. While it may adjust the listing based on the user’s food preferences and previous dining experiences, the user must still choose an appropriate restaurant from the list by taking into account factors not directly related to dining. For example, if the user’s purpose in visiting a restaurant is to meet some friends, then restaurants near the campus are more appropriate. Moreover, if these friends are on their way to a class, the top priority may be the proximity to a subway stop so that the friends can quickly travel from the restaurant to their class after lunch.

In order to develop a context-aware application, there needs to be a framework to allow the design of context-aware applications, in order to embed some intelligence into the application.

However, in order to have a context-aware application it needs a source for context, this is addressed in the next section.

1.4 Collecting context

The entities used for collecting the data that later will be transformed into context information are called sensors. A sensor may be a physical device or implemented in software. For measuring temperature some kind of physical thermometer is required, such as a thermistor, connected to an analog to digital converter interfaced to a computer and with suitable driver software. For the purpose of this project, we will use a mobile device as the sensor.

The mobile device of tomorrow will see what the user sees, hear what the user hears, and learn patterns in the user's behaviour. This will enable the device to make inferences regarding whom the user knows, whom the user likes, and even what the user may do next. This is what we need to collect and segment our context information.

We could find one example of this in The Reality Mining project [5] that has carried out a study of mobile phone usage patterns for 100 people over an extended time period. Their results provide insight into both the users as well as the ease of use of the device itself. The most important result for this thesis is that their Hiden Markov Model [6] was able to provide a good separation of office, home, and elsewhere (in their case) considering both the hour of the day as well as whether it was a weekday or weekend. That lets to know where the user is, using the user's device, so collect his context information. This is important because it concerns the automatic identification of structure in routine, and lets the system both detect and label different contexts.

It is important to note that the data delivered the mobile device needs to be interpreted, managed, and expressed in some way to be useful for applications. Furthermore, the context information may need to be stored somewhere and there must be some mechanism for exchanging context information between devices. The information relevant to a device could be stored locally or remotely. If the former approach is used, popular information will be duplicated, but this offers greater personal integrity. On the other hand, if the latter is used more communication is required and there is a major decrease in personal integrity.

In the next sections, an introduction is given to each of the technologies that this project takes advantage of.

2. Background

This chapter gives a general introduction to the Session Initiation Protocol (SIP), the SIP Express Router (SER), a brief introduction to XML, the Call Processing Language (CPL), Resource Description Framework (RDF), and finally the Web Ontology Language (OWL).

It is also important to mention the previous work of Oukhay Younes and Alisa Devic in their respective theses. Younes's thesis [7] provides a very useful background to understand most of the concepts used in my thesis. Similarly Devlic's work [8], on Semantic agents for location-aware service provisioning in mobile network, and with her subsequent report, about CPL extensions [9], forms the basis for my theses. My theses project would have been impossible without this earlier work and its documentation.

2.1 Session Initiation Protocol

2.1.1 Basic concepts

The Session Initiation Protocol (SIP) is an application layer protocol for creating, modifying, and terminating sessions with one or more participants. Each participate is located at an endpoints and is using a User Agent (UA). These sessions include Internet telephone calls (VoIP), chat, video, interactive games, or virtual reality. SIP was initially released in 1999 as RFC2543 [10] and has evolved to RFC3261 [11] and many subsequent revisions and extensions.

SIP is a text-based protocol, similar to HTTP and SMTP, and follows the internet design philosophy by utilizing intelligence in the endpoints, rather than providing services in the network. The SIP user agent client (UAC) generates and sends requests to a user agent server (UAS). SIP is designed to be generic and is independent of the underlying transport protocol and type of the session. It is designed to run on top of UDP but can utilize TCP or SCTP or TLS/SSL.

One of the strengths of SIP is that the user only needs a single address called a Uniform Resource Identifier (URI). As SIP provides a mechanism for mapping this URI to the user's current device and point of network attachment it supports user mobility. We will talk more about this URI in section 2.1.3.

2.1.2 SIP communication example

To understand how a SIP session is established we will explain a simple example shown in Figure 2. Suppose that Alice wants to call to Bob. Both use a SIP capable device (i.e., a device with a SIP user agent). Alice enters Bob's SIP identifier (URI). Alice's user agent (UA_1) has to determine the location of the remote end point, currently being used by Bob.

In order to do so UA_1 sends an INVITE message to its SIP proxy. Alice's SIP proxy uses the DNS system to learn the IP address of Bob's SIP proxy. Alice's SIP proxy returns the state of enquiry (100 Trying), and the Bob's proxy contacts the remote user agent (UA_2) indicating that there is an incoming call from Alice. UA_2 starts ringing (180 Ringing) and this information will be transmitted to Alice's UA (UA_1).

If Bob accept this call, UA_2 will send a message (200 OK), via the proxies of both parties. Alice's user agent (UA_1) will send an acknowledge message (ACK) to Bob's user agent (UA_2), from this point on the two user agents will directly exchange the mutually agreed upon media data in a media session.

To end the session, one of the user agents sends a BYE message to the other, the other party will reply with an OK (200 OK).

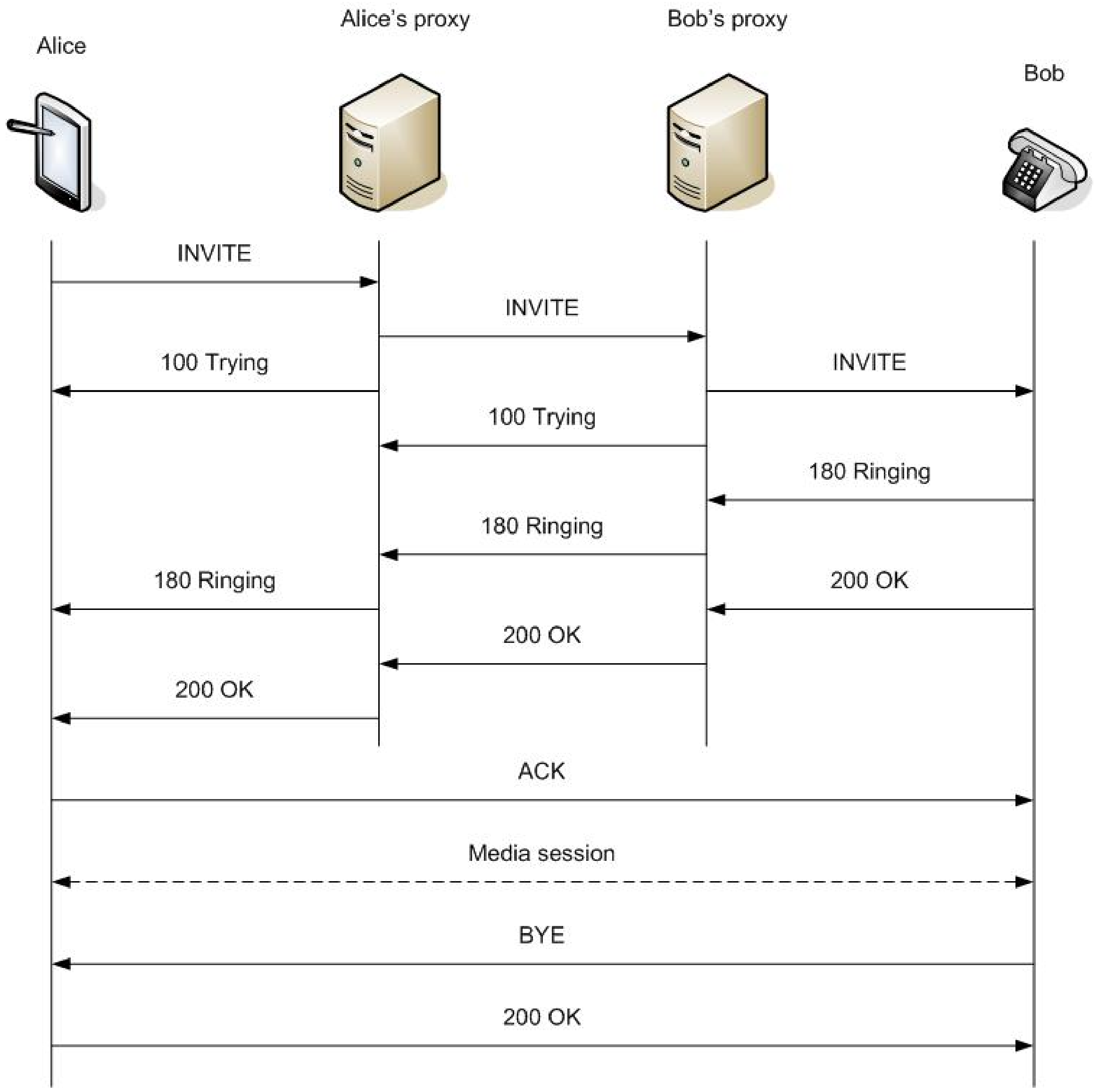


Figure 1: Simple example of two parties using SIP

We note that only the SIP signaling messages are transferred via proxies, while all the session media is transferred directly between the peers.

2.1.3 SIP URI

As briefly discussed above, the SIP URI identifies the communication parties. The general format of a SIP URI is:

sip: [user:passwordd@] hostname | ipv4addr | ipv6addr [:port; uri-parameters]

The token "sip:" specifies that the URI is a regular SIP URI. There are two types of URIs in SIP, namely SIP and SIPS. The difference is that SIPS provides additional security. When SIPS is specified, then the path from the initiator to the target should be secured. In practice this means that either Transport Layer Security (TLS) [12] or Secure Sockets Layer (SSL) [13] is used to tunnel the data hop-by-hop.

- **User**:	The identifier.
- **Password**:	The user's password. It is not recommended to use this field due to the fact that it is sent in clear text (unless SIPS is used).
- **Hostname**:	An IP address (IPv4 or IPv6) or domain name.
- **Port**:	The port number where the request should be sent. The default port will be used if is not specified (5060 for SIP and 5061 for SIPS).
- **Uri-parameters**:	An arbitrary number of parameters to the request, e.g., transport = tcp.

Some examples of SIP URIs:

sip: 3864459@139.43.19.1
sip: telephony@telecom33.com
sip: user:password@139.43.19.1
sip: +34-93-3000000@gateway.upc.es;user=my_telephone

2.1.4 SIP Message structure

SIP messages consist of a request initiated by a client and responses from servers. The syntax and the structure of the header has the same format as in the HTTP protocol. The syntax of the messages is a header and an optional body.
SIP uses six types of requests, also called methods (see section 2.1.5) [14].

To understand how a message in SIP is constructed we will look at an example INVITE.

Note that only the most common fields are discussed.

```
INVITE sip:bob@example.com SIP/2.0
Via: SIP/2.0/UDP pc234.example.com:5060;branch=z9hG4bK776asdhds
To: Bob <sip:bob@example.com>
From: Alice <sip:alice@example.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc234.example.com>
Content-Type: application/sdp
Content-Length: 142
```

Figure 2: INVITE message

The first line specifies the method or request type (INVITE in this case). The INVITE is followed by the address destination and finally the SIP version number that the caller supports.

- **Via**:	Is inserted by the User Agent which initiated the request. Shows the path the request has taken in the SIP network, thus allowing responses to be routed back the way the request came. It specifies the transport protocol, the client's host name or network address (and optionally port number), and a branch value that is used to loop detection.
- **To**:	Specifies the address of the recipient of the message.
- **From**:	Contains the address of the initiator of the request.
- **Call-ID**:	Identifies the messages that are part of a conversation. It is unique for this call. The identifier is generally based upon a random number and the host name or IP address of the UA.
- **CSeq**:	The Command Sequence number (CSeq) is used to order transactions within a dialog and can be used for differentiating requests and retransmissions. It is incremented for each new request within a dialog. It is initialized at the start of the call. When an ACK message or a CANCEL message is sent as a response to an INVITE message, the CSeq number is the same as in the INVITE message.
- **Contact**:	Specifies a SIP URL that points directly to the sender, for direct communications between user agents.
- **Content-type**:	The media type of the message body. It could be text/plain, application/cpl+xml, etc.

- **Content-length**: Indicates the size in of the message body in bytes.

2.1.5 SIP request messages

The SIP request messages are created by a User Agent Client (UAC). Those messages are sent to a SIP server, who takes the appropriate action(s) depending on the content and type of the request. Always, except for the ACK message, requests are followed by a response.

- **INVITE**: An INVITE is sent by a UAC to initiate a SIP session. This INVITE message will contain a session description protocol (SDP) description that describes the media, the address of the caller and the address of the addressee, the user location, and the CODECs to use. After the three-way handshake the session is established and completed, where the INVITE is the firs message to send. Details of SDP can be found in [15].

- **ACK**: The ACK message confirms that the response to the INVITE request, namely 200 OK. It is sent by the UAC.

- **OPTIONS**: The OPTIONS message can be used for discovering information about another UA or a proxy. It is not used to establish communications.

- **BYE**: BYE is used by the user agent to indicate to the server that wants to terminate an ongoing session.

- **CANCEL**: CANCEL is used to cancel a session before it is established or to cancel a previous request. If a final response for the request already has been received, the CANCEL message will have no effect. CANCEL requests can be sent by both proxies and UACs.

- **REGISTER**: This request allows the user to register the address of a UAS with a SIP server. The address in the “To” header will be associated with a user.

2.1.6 SIP response messages

The response to the SIP messages interprets a request, and indicates success, failure, or provides the client with more information that may make the request successful. Also, they can send the state of the server.

There are a lot of different response messages in SIP and they are grouped into six different types. Each response is codified with three digit number where the first digit is the response type [16].

- **1xx: Informational**: The request was received by the server, but further processing is required. This message should be sent if the request is expected to take longer than 200ms to process.

- **2xx: Success**: The action was successfully received and accepted.

- **3xx: Redirection**: Gives information about the callee´s new location or if the call wasn't successful or if the address was correct, possible alternative services.

- **4xx: Request Failure**: The client sent a request that contains bad syntax or cannot be fulfilled at this server. The client should modify the request or try another server.

- **5xx: Server Failure**: The request was valid, but the server was unable to process it successfully.

- **6xx: Global Failure**: The contacted server has enough information to claim that the request cannot be fulfilled anywhere in the system.

Table 1 gives some examples of SIP response message:

Status Code	Meaning	Status code	Meaning
100	Trying	500	Not Implemented
180	Ringing	503	Service Unavailable
200	OK	504	Server Timeout
301	Moved Permanently	505	Version Not Supported
302	Moved Temporarily	600	Busy Everywhere
400	Bad Request	603	Decline
401	Unauthorized	604	Does Not Exist Anywhere
405	Method Not Allowed		
408	Request Timeout		
411	Length Required		
415	Unsupported Media Type		
482	Loop Detected		
486	Bust Here		
489	Request Terminated		

Table 1: SIP response codes and their meaning

2.1.7 Register

The purpose of registration is to establish the user's current device and point of network attachment (i.e., IP address). To do so, each UA needs to register its contact URI at Registrar server. Usually, this is the first thing that is done when a SIP enabled device comes online.

Registration is initiated by the UAC by sending a REGISTER request. If the registration is accepted the server will store a binding between the URI and the contact address. For that reason, REGISTER request includes one or more Contact headers. The Registrar server will respond with a 200 OK message if the registration was successful. The same REGISTER request message is used when the user wants to update an address, get the currently addresses stored on the server, or delete an address.

Only the location server needs to know the user's current location. This has significant privacy implications. The location server can apply various polices to decide who can learn the user's location, i.e., a user's location server can decide who can ask for the user location and when they can ask (perhaps even limiting it to where they can ask from). This scales well, as only the user agent client has to update their own location server, rather than having to inform all possible callers.

The registration will expire after a certain time, as specified by the local policy. However, the client can suggest an expire interval in the REGISTER request. Because a binding is deleted when the registration expires it has to be renewed. The client learns the current expiration interval in the 200 OK message from the registrar server. It is up to the client to renew the bindings before they expire, if the client wishes to be able to receive incoming INVITE requests. Before the REGISTER message can be sent the Registrar server has to be located. Several alternatives exist, such as manual configuration, DHCP, or DNS SRV Resource Records. Note that DHCP can only be used if the user is in the domain which is also their SIP domain.

2.2 SIP Express Router

SIP Express Router (SER) is a high-performance, configurable, free SIP server. It can act as SIP registrar, proxy or redirect server. SER also provides presence and messaging, RADIUS/syslog authorization and authentication, translation between SIP and SMS or Jabber, etc. SER can be configured for many scenarios including small-office use, enterprise PBX replacements and carrier services. SER is publicly available under the terms of the GPL license [17].

SER was initially developed by Fraunhofer Fokus in 2001. Part of the Fokus team then moved, with the SER copyrights, to a newly created company, iptel.org [18] in 2004. Two of the other SER core developers and one main contributor started a new Open Source project called OpenSER [19]. Additional open source code was contributed by independent third parties.

SER is built around a processing core that receives SIP messages and provides the basic functionality for handling SIP messages. Most of SER's functionality is offered through its modules, much like the Apache web server. By utilizing a modular architecture, SER is able to have a core that is very small, fast, and stable. SER modules expose functionality that can be utilized in the SER configuration file, ser.cfg. The ser.cfg configuration file controls which modules should be loaded and defines how the modules should behave by setting module variables.

We have chosen SIP Express Router to upload and execute CPL scripts because is a scalable and reliable SIP platform [20] that supports CPL; and because it was previously used in Alisa Devlic's work (thus I can build directly upon her work).

2.3 Extensible Markup Language

XML [21] stands for eXtensible Markup Language and became a W3C [22] Recommendation in February 1998. XML is a text format derived from SGML [23]. It is a markup language like HTML and was designed to describe data. It is important to note that XML is not a replacement for HTML. XML and HTML were designed with different goals: XML was designed to describe data and to focus on what the data is, while HTML was designed to display data, with a focus on how data looks. Therefore, HTML is concerned with displaying information, while XML is concerned with describing information.

XML was not designed to do anything itself, rather XML was created to structure, store, and to encode information. The following example is the title of the thesis, stored as XML:

```
<thesis>
        <title>Model driven context awareness</title>
        <author>Sergi Laencina Verdaguer</author>
</thesis>
```

As can easily be seen a **thesis** has a title and a author. But still, this XML document does not do anything. It is just pure information wrapped in XML tags. Someone must write software to send, receive, or display this information, but the XML make it easier for them to know what the information is. XML facilitates data sharing between different platforms, but can also serve as a base for creating specialized markup languages such as XHTML, RDF (see section 2.5), OWL (see section 2.6), MathML, MusicXML, etc.

Note that tags are not predefined, hence an author must define his or her own tags. Elements are data containers that are defined by their name; they may also have attributes and content. The tags used to mark up HTML documents and the structure of HTML documents are predefined. The author of an HTML document can only use tags that are defined in the HTML standard, but XML allows an author to define his own tags and his own document structure. XML has a strict syntax. We say an XML document is well formed if its syntax respects the XML specification constraints. For instance, each opened tag should be closed.

XML uses a Document Type Definition (DTD) or an XML Schema to describe the data. If a document is well-formed and compiles with a schema (DTD or XMLSchema), it is said to be valid. To determine if a document is well formed and valid we use software called “parsers”. Validity is hence a stronger constraint than simply being well formed. A schema specifies a number of constraints on the structure and content that go beyond the basic constraints imposed by XML itself. Therefore XML is designed to be self-descriptive XML [24].

2.4 Call Processing Language

The Call Processing Language (CPL) is a language designed to describe and control Internet telephony services. It is used to write scripts for call handling. It is an IETF standard described in RFC3880 [25]. It is also an XML based language. It works on top of SIP or H.323 [26]. It can be implemented on either network servers or user agents; as both can usefully process and direct users' calls. Its greatest advantage is its simplicity, its extensibility, and its independency from the underlying of operating system or signaling protocol. It was developed for end user service creation and is purposely limited its capabilities. It is a safe language for non-experienced users, as it can only access limited resources, cannot call external programs, and does not provide loops or recursion.

As any XML based language, CPL has a DTD that specifies its syntax. Therefore, CPL scripts can and should be validated before they are uploaded to a server. Note that the CPL standard does not specify a how to upload a script. However, this upload can be realized in a secure manner. So only the user can redirect or process their calls we require that the uploading mechanism provide both confidentiality of the request and also authentication.

A CPL script specifies call processing actions. We can define two types of call processing actions: top-level actions and subactions. Top-level actions are actions that are triggered by signaling events that arrive at the server. There are two top-level actions defined: "incoming", the action performed when a call arrives whose destination is the owner of the script, and "outgoing", the action performed when a call arrives whose originator is the owner of the script. Subactions are actions which can be called from other actions.

CPL enables the SIP server to provide many features, such as call blocking, call redirecting, call forward on busy/no answer, intelligent user location, etc. These functions can be applied either outgoing or incoming calls. CPL can be used to specify quite complex features. We can have a script which performs the following:

> When a caller tries to call Bob’s desk phone, if Bob does not answer in 10 seconds, then calls from Bob’s boss are forwarded to Bob’s mobile phone, and all other calls are directed to Bob’s voicemail.

The scripts used in this project are included in appendix A.

2.5 Resource Description Framework

The Resource Description Framework (RDF), developed by World-Wide Web Consortium (W3C), provides support for describing and exchanging metadata on the Web. Through the Uniform Resource Identifier (URI) it allows a description of Web resources to be made publicly available in a machine understandable form.

The motivation for introducing RDF in the Semantic Web [27] came from limitations of XML markup language. As XML does not provide any means of talking about the semantics (meaning) of data and RDF is essentially a data-model. Its basic building block is an object-attribute-value triplet, called a statement.

Of course, an abstract data model needs a concrete syntax in order to be represented and transmitted, and RDF has been given a syntax in XML. As a result, it inherits the benefits associated with XML. Although, syntactic representations of RDF, not based on XML, are also possible.

RDF describes resources through a collection of properties, called an RDF Description, each of which consists of property type and value. Resources can be thought of as an object that can be described. Resources can be authors, books, publishers, places, people, hotels, rooms, etc. Each resource is assigned URI. This URI can be an URL (Uniform Resource Locator, or Web address) or some other identifier that does not necessarily enable access to the resource.

An RDF document consists of statements, each of which is a combination of a resource, a property, and a value assigned to that property. Thus RDF statements are written in triples: subject, predicate, and object. RDF provides semantics and describes relationships between objects. An example of a statement is:

Student X is the author of the master thesis Y

The easiest way of interpreting this statement is to use the definition and consider the triple:

(“Student X”, http://www.mydomain.org/author,
http://www.example.edu/YmasterThesis.pdf).

The triple (x, P, y) can be thought of as a logical formula P(x, y), where the binary predicate P relates object x to object y. RDF offers only binary predicates (properties). Note that the property "author" and one of the two objects are identified by URLs, whereas the other object is simply identified by a string.

RDF doesn't have a built-in mechanism for defining properties and describing relationships between resources and properties. That is the task of RDF Schema (RDFS).

Contrary to XML Schema, that simply constrains the structure of XML documents, an RDF Schema defines the vocabulary used in RDF data models. In an RDFS we can define the vocabulary, specify which properties apply to which kinds of objects and what values they can take, and describe the relationships between objects. Additionally, this enables checking an RDF resource's validity just as with an XML Schema [28].

2.6 Web Ontology Language

The Web Ontology Language (OWL) [29] is a markup language for publishing and sharing data using ontologies on the Internet. The expressivity of RDF and RDF Schema, defined as a measure of the range of constructs that can be used formally, flexibly, explicitly, and accurately to describe the components of ontology, is very limited. Therefore, OWL is a vocabulary extension of the Resource Description Framework (RDF). Together with RDF and other components, these tools make up the Semantic Web project [30].

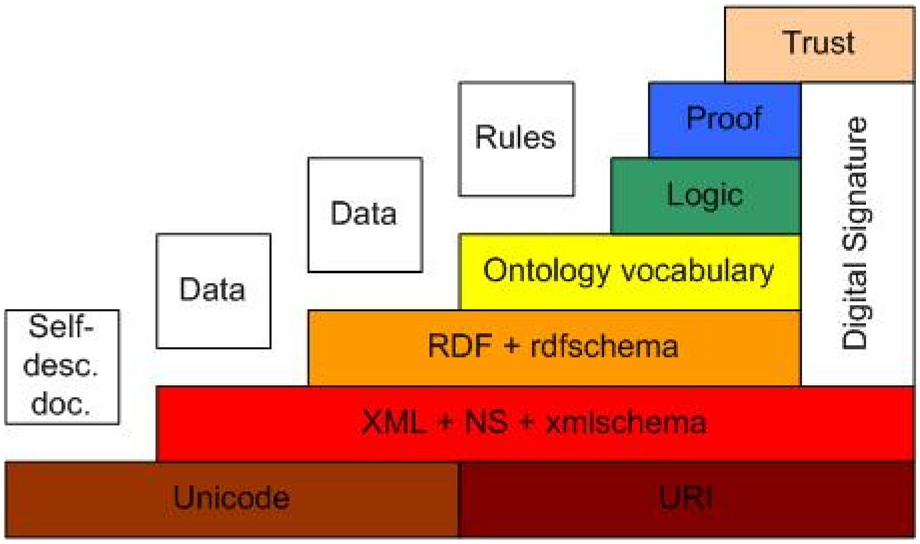


Figure 3: A Semantic Web stack from Tim Berners-Lee [31]

OWL represents the meanings of terms in vocabularies and the relationships between those terms in a machine-accessible way.

DAML+OIL was a starting point for the W3C Web Ontology Working Group in defining OWL. It was a joint initiative of a number of research groups from the United States and Europe to define a richer language. The name is a conjunction of the U.S. proposal DAML-ONT [32] and the European language OIL. OWL was developed as a vocabulary extension of RDF and is derived from the DAML+OIL Web Ontology Language. It is a language that its authors hoped would be broadly accepted to provide standardized ontology language for the Semantic Web

OWL was designed specifically to provide a common way to process the content of web information. The language is intended to be read by computer applications instead of by humans. Since OWL is written in XML, OWL information can be easily exchanged between different types of computers using different operating systems and programming languages. OWL builds upon RDF and RDF Schema because a XML and RDF syntax is used, instances are defined using RDF descriptions, and most of the RDFS modelling primitives are used.

OWL provides three increasingly expressive sublanguages designed for use by specific communities of implementers and users.

3. Method

3.1 Case study

The purpose of this thesis is to combine some of the earlier voice over IP work with context awareness for mobile users in a demonstrator for visitors to Kista. The first step is a demonstrator for students so that their incoming or outgoing calls can be processed differently depending on the user's position, task, or activity. For example, if a student is in class, his device should detect that the student is in Kista (establishing a location), that he is in a class (establishing his current task), and he is listening (establishing his current activity). The device sends a message to the student's SIP proxy, reporting all of this information. If there is an incoming call for the student from his friend, the SIP proxy automatically rejects the call because the student is busy. However, if the student has a break in their class, now his activity is "break", so the device informs the SIP proxy of this break status, now if there is an incoming call, the student can answer it.

To implement the above behaviour we need to know the context of the user and then establish an appropriate set of actions.

3.2 Implementing Context awareness

In order to implement this context awareness, we need to create a suitable set of CPL scripts, which will be stored at the user's SIP proxy. Next we have to learn the user's routine and what actions he might want to take in each context. We begin by exploiting location, thus based upon where the user is and where they are expected to be going we can associate different CPL scripts (i.e., preferred actions). In order to increase the "correctness" (i.e., selecting the action that the user would most prefer in this context) we can use additional context information. The first additional parameters which we will consider are the user's current task and activity. In order to decide what the user's current task and activity are we need to infer this information from various sensor data, and we will do this using OWL. Thus when an incoming call arrives at the SIP proxy it can reason about this call using the user's current context to decide what action should be taken for this call.

Note that we have treated location as a special case of context information, not only because this has been used previously, but also because there are several other students currently developing techniques to provide better location information about Kista (both indoors and outdoors). These students are both using modelling techniques (outdoors and indoors) and using new signal processing techniques at indoor access point [33] and [34].

To learn the user's routine, the method which we will use to define and detect the user's routines as well as establish what the user's preferred actions would be, will be based upon "blogs", specifically the special version called life blogs.

For example, we can look at companies such as Google, MSN Search, and Yahoo that promote theirs respectively Google's Blogger, MSN Spaces, and Yahoo's Blog Directories instead of only queries on cached web content. They know that there are many bloggers that wants to transcribe and publish the minutiae of their lives. However, there are a lot of people who don't take the time or don’t want to write such diaries. So, we have to think how to get that information, for example creating a life blog for the user.

4. Analysis

In this part of the thesis we try to define a routine for a user. Given this information, if we understand the user's preferred actions, then we can define easily how we should act. Therefore, we need to detect, extract, and if possible predict the user's routine.

Given the above information, we need to define how we will collect that information and from it derive useful context information, we must also decide where we are going to process this information and where we will store it.

4.1 Life Log

In order for our context awareness system to know a user's routines, tasks, and preference, we will mine the user's life blog. We can then use knowledge of these routes to anticipate what the user is going to do next, thus we can anticipate what actions the user will take – perhaps based upon what the user did in this or a similar environment last time. We begin by examining the typical routine of a student that is doing his master theses and also taking a course.

The structure of the "blog" is the following: First we write the task and/or where the user is. For each task or place we try to anticipate which calls the user wants to receive and how we can know where the user is.

A student Blog of a daily routine in Kista (In parenthesis the approximate time of the start of some specific task).

- Leave home (8:30)
    - Incoming calls the user want to receive: Any
    - How can we know where the user is: The user misses the signal of his Access Point (AP). Also, we can know when the user leaves home looking at his timetable.

- Home Tunnelbana (8:35)
    - Incoming calls the user want to receive: Any
    - How can we know where the user is: We suppose that there the user doesn't have wireless connection, only GPRS or 3G. Also, the mobile phone can knows the timetable of the underground, so it can know in which train the user is travelling.

- Travel to Kista (8:40 – 9:10)
    - Incoming calls the user want to receive: Any
    - How can we know where the user is: We suppose that there the user doesn't have wireless connection, only GPRS or 3G. Also, the mobile phone can knows the timetable of the underground, so it can know in which train the user is travelling.

- Kista Tunnelbana (9:10)

- Incoming calls the user want to receive: Any
    - How can we know where the user is: The user detects the AP from the Kista station. Also, we can know with the information of the underground timetable, in which train the user has arrived. Now, the user has wireless coverage.

- Travel from Kista Tunnelbana to wireless@kth (9:10 – 9:20)
    - Incoming calls the user want to receive: Any
    - How can we know where the user is: Depending on which APs the user will find during the travel. If the user loses the wireless coverage, he has to connect to GPRS or 3G.

- Arriving at Lab (9:20)
    - Incoming calls the user want to receive: Any
    - How can we know where the user is: The user detects the AP of the lab.

- Take a coffee (9:25)
    - Incoming calls the user want to receive: Friends
    - How can we know where the user is: The user detects the AP of the wireless@kth or the “coffee AP”.

- Lab (9:25 – 10:30)
    - Incoming calls the user want to receive: Emergency + Family
    - How can we know where the user is: The user detects the AP of the lab.

- Take a coffee (10:35)
    - Incoming calls the user want to receive: Friends
    - How can we know where the user is: The user detects the AP of the wireless@kth or the “coffee AP”.

- Lab (10:35 – 12:00)
    - Incoming calls the user want to receive: Emergency + Family
    - How can we know where the user is: The user detects the AP of the lab.

- Bathroom (12:05)
    - Incoming calls the user want to receive: Emergency
    - How can we know where the user is: The user detects the AP of the wireless@kth or the “Bathroom AP”.

- Lab (12:05 – 13:00)
    - Incoming calls the user want to receive: Emergency + Family
    - How can we know where the user is: The user detects the AP of the lab.

- Travel from wireless@kth to lunch (13:00 – 13:15)
    - Incoming calls the user want to receive: Any
    - How can we know where the user is: Depending on which APs the user will find during the travel. If the user loses the wireless coverage, he has to connect to GPRS or 3G.

- Lunch (13:15 – 14:15)
  - Incoming calls the user want to receive: Any
  - How can we know where the user is: The user detects the AP of the restaurant.

- Travel from lunch to wireless@kth (14:15 – 14:30)
  - Incoming calls the user want to receive: Any
  - How can we know where the user is: Depending on which APs the user will find during the travel. If the user loses the wireless coverage, he has to connect to GPRS or 3G.

- Lab (14:30 – 15:30)
  - Incoming calls the user want to receive: Emergency + Family
  - How can we know where the user is: The user detects the AP of the lab.

- Bathroom (15:35)
  - Incoming calls the user want to receive: Emergency
  - How can we know where the user is: The user detects the AP of the wireless@kth or the "Bathroom AP".

- Lab (15:35 – 16:45)
  - Incoming calls the user want to receive: Emergency + Family
  - How can we know where the user is: The user detects the AP of the lab.

- Travel from wireless@kth to class (16:45 – 17:00)
  - Incoming calls the user want to receive: Any
  - How can we know where the user is: Depending on which APs the user will find during the travel. If he loses the wireless coverage, he has to connect to GPRS or 3G.

- Class (17:00 – 18:30)
  - Incoming calls the user want to receive: Emergency
  - How can we know where the user is: The user detects the class AP. Also the user can know that he is in class using his daily timetable, but what happens if the class does not start at time? One solution could be using SNMP for detect that in class there are more people than the user. If it happens, the system can suppose that the class has started. But how many people?

- Class break (18:30 – 18:45)
  - Incoming calls the user want to receive: Any
  - How can we know where the user is: Using the timetable. But if the user stays inside the class during the break, how can we detect it?

- Class (18:45 – 20:15)
  - Incoming calls the user want to receive: Emergency
  - How can we know where the user is: The user detects the class AP. Also the user can know that he is in class using his daily timetable, but what happens if the class

does not start at time? One solution could be using SNMP for detect that in class there are more people than the user. If it happens, the system can suppose that the class has started. But how many people?

- Travel from class to Kista Tunnelbana (20:15 – 20:30)
    - Incoming calls the user want to receive: Any
    - How can we know where the user is: Depending on which APs the user finds during the travel. If he loses the wireless coverage, he has to connect to GPRS or 3G.

- Kista Tunnelbana (20:30)
    - Incoming calls the user want to receive: Any
    - How can we know where the user is: The user detects the AP from the Kista station. Also, we can know, with the information of the underground timetable, which train the user will take. He has wireless coverage.

- Travel to home (20:35 – 21:05)
    - Incoming calls the user want to receive: Any
    - How can we know where the user is: We suppose that there the user doesn't have wireless connection, only GPRS or 3G. Also, the mobile phone can knows the timetable of the underground, so it can know in which train the user is travelling.

- Arrive at home (21:05)
    - Incoming calls the user want to receive: Any
    - How can we know where the user is: The user detects the coverage of my AP. Also, with the timetable, approximately.

It is important to note that for outgoing calls, in principle, there isn't problem. The user might specify a budget and thus the system may need to decide which interface (GPRS or WLAN) and even if a call should be permitted in a given setting (for example, in a non-emergency setting in a class or theatre; or based in the expected cost of the call vs. the user's budget). However, for the rest of this thesis we will focus only on incoming calls as we want to remove load from the user by automating the decision of what should be done with each incoming call.

So, how can we know where the user is? This can be based upon:

- Time (using the user's personal schedule).

- Place (We might know where the user is because we can recognizes a specific AP whose location we know, or perhaps we can use the location of several APs and their signal strengths,…).

- Social Context (Based upon the other people that are with the user we might be able to recognize that the user is in a particular class, or a meeting, or in the cafeteria).

Social context can be a very powerful cue, since if the user is in the lab working with his thesis advisor, the user might only want to be disturbed by an emergency call. Thus if we can detect this context, we can automatically apply the user's selected rule

Using an ontology we can easily create rules that apply to a group of people; for example, students, friends, family, etc. This avoids the user having to specify the same rule again and again for different people.

Ideally we would like to be able to automatically determine the user's preferences based upon observing which calls they accept and reject. For example, when I am taking a coffee, I only want to receive calls that are in my contact list, because I prefer to spend this short time talking with them.

So, what happens if the user changes his routine, how we can detect it? And also, if we don't forward calls to the user, how would the user know about something that he or she is purposely not being made aware of? This suggests that either we need a way for users to see what the results of the system decisions were and to note which were incorrect.

However, we will begin with the problem of extracting, detecting, and predicting what the user's behaviour patterns are.

4.2 Detecting, extracting, and predicting user behaviour

It is commonly known that although humans have the potential for an extremely high degree of randomness, there exist easily identifiable patterns in every person's life. These can be found on a range of timescales: from the daily routines of getting out of bed, eating lunch, and travelling home from work; to weekly patterns such as Saturday night go out; to yearly patterns like seeing family during Christmas. Many of these patterns in behaviour are easy to recognize; however, some are more difficult. But one solution could be to use data from mobile phones to create a system that can accurately perceive and predict actions in a user's life.

We attempt to quantify the amount of predictable structure in an individual's life using an information entropy metric, understanding entropy as the measure of the disorder in a system.

Therefore, people who live high entropic lives tend to be more variable and harder to predict, while low-entropy lives are characterized by strong patterns across all time scales.

Calculating life's entropy can be used as a method of self-reflection on the routines (or ruts) in one's life, but it can also be used to compare the behaviours of different demographics.

4.2.1 Detecting user behaviour

The phone application itself can help to detect the user behaviour. One example could be: If a user spends a significant amount of time in the range of a specific wireless coverage zone, the Context application makes the phone vibrate and prompts the user to name the location or situation. Examples of user input names include "Wireless Lab," "My Dorm," "Bob's Apartment," "Restaurant Electrum", and so on. And all of this information will be used to predict the user behaviour.

Another option would be utilizing a simple labelling application, which when the user wants to establish a labelled action it would listen for what it hears and then record the time, list of what it hears (i.e. the signal strengths), and the user's choice of action(s). Also, collect data at the SER and then show what actions are involved when the user is in this context again. This is part of the Behdis Zandieh thesis [35] where she is creating three sample applications: a scanner to collect the data and build a 3D model, a location finder (a simple location based service), and a bandwidth advisor that is an example of a model based service.

4.2.2 Extracting user behaviour

Labels for locations can be learned from the phone application itself, but are also input by the user through an application interface. We need to know the user behaviour. A first and easy solution could be to write a life log of the user, what we have done in the previous section. This could be creating an interactive and automatically generated diary application that enables users to query their own life: When was the last time I lunched with Bob? Where were we? Who else was there? When did I get home? This allows knowing exactly the user's routine, and using that information to understand the user's behaviour. We segment the life log and recognize the routine.

When we say that segment the user's life log, we mean that we create different states, like working, in class, lunching, at home…, so then we can apply the different actions to every state.

But also, we need to visualize the model's predictions about upcoming behaviour in the immediate future to anticipate to the user's actions

4.2.3 Predicting user behaviour

In our work, we defined the user routine. We define this in terms of the user's actions and we use that information to create our system. But, what happens if the user changes his routine, how we can detect it? We can know using the life log where the user must be in a certain time. For example, if we know that the user has just finished off lunch and is not yet at class, we can deduce that he is travelling. So, we can use the knowledge of these routes to anticipate what the user is going to do next.

This prediction based on where the user must be, allows us to identify the user’s location and anticipate to his or her actions to provide a better service.

4.3 OWL and context aware services

We use the CPL scripts to specify call processing actions. We can block, redirect, accept,… calls. And all of these functions can be applied to either outgoing or incoming calls. But we need to define when a CPL script must me used instead of others. And here is when appear the context and the ontologies, which define the current task, activity and location of the user.

4.3.1 Using ontologies

The general paradigm for using ontologies in context aware services is to define an intermediary semantic layer between context sources and context users. As the sensors provide raw context items, we need some “adaptor” to format these context items into OWL context items that can directly feed the OWL context database. Similarly, we may define some context providers that can handle context information queries issued by context aware services. These context providers are in charge of querying the database and answering queries so as to provide services with high level context information. The context providers can take advantage of technologies such as OWL-S [36] for service invocation for example. The following figure summarizes this:

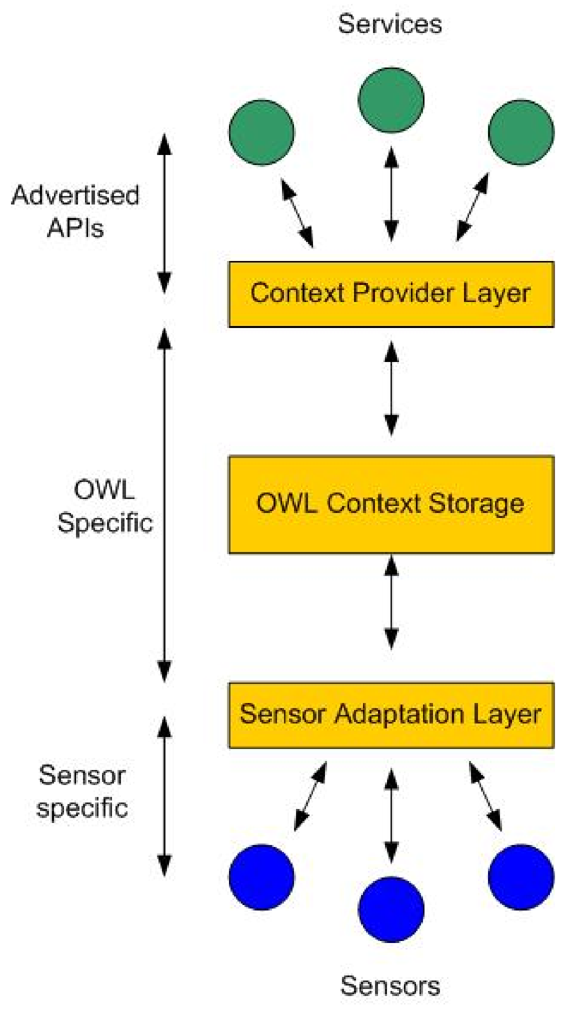


Figure 4: The ontology architecture paradigm

Note that with these context providers, any service can query the context database, as it is an open architecture. Context providers can also implement access control mechanisms. This is necessary when users wish to protect their context data. One might not want others to know where you are and what you are doing. A very simple policy that does not require any management is: any user can access all context concerning himself and only context concerning himself. A second approach based on the first, would enable user A to define which context items he wishes to share with whom. For instance, the user Bob lets his boss (another user) know about his location (context item) during working hours.

This pattern can then be extended. In our implementation, we simply use an OWL file for context storage. The OWL file is then parsed to extract the relevant data we would like to use.

This case is the most straightforward one, but we can subsequently take advantage of the use of a knowledge database to store our context information. Such a knowledge database has many interesting features. First, it makes reasoning on context much easier.

Secondly it can prevent (or detect and correct) inconsistencies in the context information it manages. Inconsistencies are contradictions in the context information stored. For example, if a user's context indicates that he is working and in the bathroom, there may be inconsistency. Knowledge databases can be configured to tolerate more or less knowledge inconsistency and to keep the coherence of the gathered information.

4.3.2 Reasoning based on context rules

We used reasoning based on context rules in our implementation. We have used a set of predefined rules to draw conclusions based upon the user's context. Here is an example:

Consider the user Alice; she has a context item that describes her location. This item is set to "Lab" and we have the predefined context based rule:

"If a user is in a working place, then he is busy, incoming calls from friends should redirected to Alice's assistant, and all other calls should be rejected"

Now, if Bob (a friend of Alice) calls he will be redirected to the assistant because we can determine that Bob is a friend and that Alice is in a working place from the reasoning based on ontologies. Often, both types of reasoning are combined as in the above example. In appendix B we can find some examples of purposed OWL scripts of our user life log.

4.4 SER, CPL, and OWL

Once we have the CPL and OWL scripts, we need to store for then be treated. One solution could be use the Alisa's Devlic work purposed in her thesis, that consists in uploaded the CPL scripts to a SIP proxy server, SER (see figure 6).

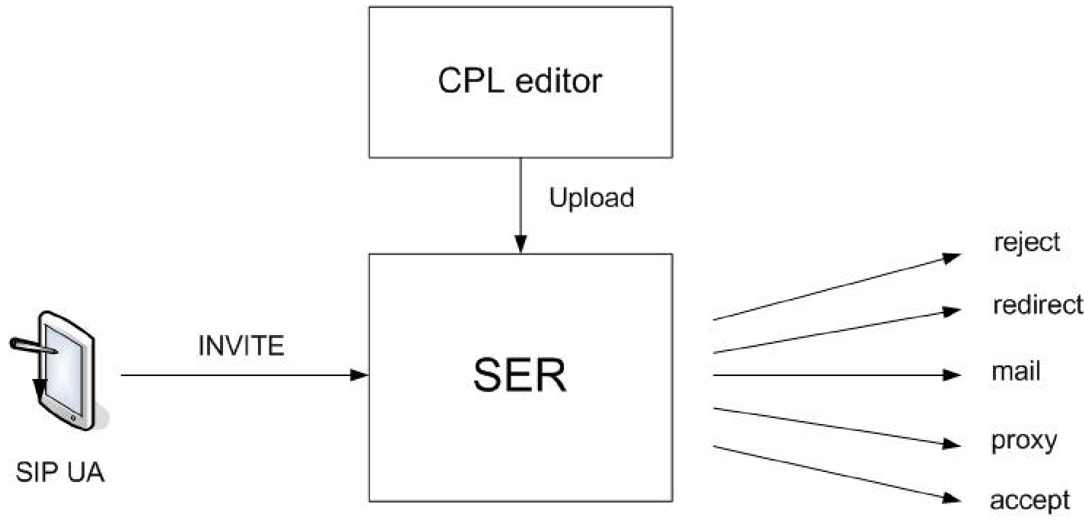


Figure 5: CPL script upload

A CPL script is parsed after uploading to SER. It is stored in an external MySQL database. When a SIP INVITE arrives (initiating an incoming or outgoing call), SER will execute the appropriate part of the user's CPL script that refers to an incoming or outgoing call and run the call routing logic (reject the call, redirect it, forward it to the voicemail, accept and route the call, etc).

CPL scripts can be uploaded using SIP's REGISTER method or with the aid of graphical programs, such as CPLEd (we can find more information of CPLEd in Oukhay Younes work [37] and Alisa's Devlic work [38]).

And the same happens with the OWL scripts. In Devlic's thesis, like in Oukhay Younes, they use a modify version of the CPLEd to upload manually the scripts. This could be a good solution, but it must be interesting in future to create one application that does the same but automatically.

It is important to note that the CPL scripts can be loaded all in the same time, but the OWL scripts will change every time that changes the user's context. In this thesis we propose a solution to this problem. We can upload the entire context or only upload changes to the OWL or CPL scripts. For this thesis, we will not implement this, but we support that the most efficient solution would be to upload only the changes.

5. Evaluation

In this section we try to evaluate a simple scenario. In our first test, we will examine a simple call, without using CPL scripts or context. This allows us to compare the responses our later more complex scenarios.

In the second, third, and fourth tests, we upload to our SIP express router (SER) a CPL script and, depending on the test, we apply different contexts.

The basic scenario is represented in the following figure:

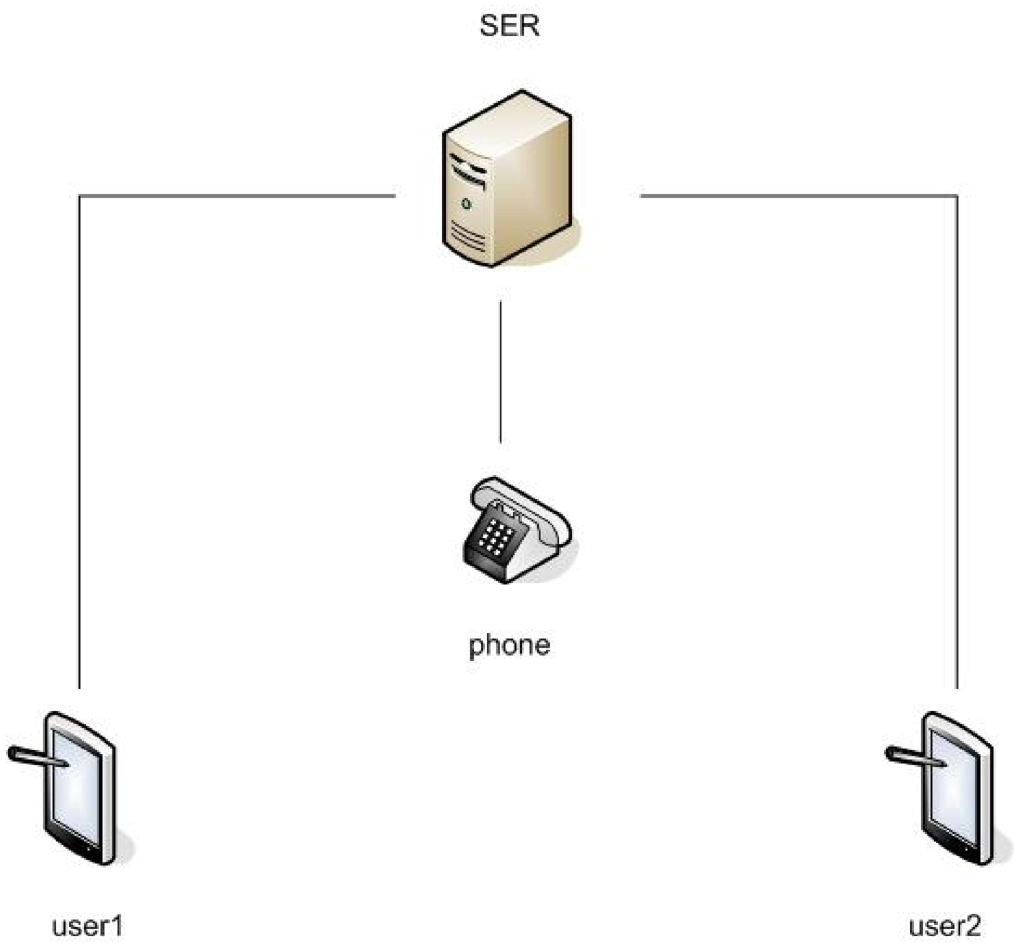


Figure 6: Network Architecture

We use:

- SER as our SIP express router.
- A user1, with the sip address user1@ccsser2.wireless.kth.se
- A user2, with the sip address user2@ccsser2.wireless.kth.se
- A SIP phone, with the sip address 1005@ccsser2.wireless.kth.se

In our SER configuration, we create the new users, user1 and user2. For the SIP user agent at user1 and user2 we use X-Lite v2.0 for Linux [39] and we configure it correctly with the appropriate user name, password, and SIP proxy (i.e., SER) address. The SIP phone is a Cisco 7960 series phone [40].

5.1 Test 1

In this test, we see a typical call between SIP user agents. Here we use the basic configuration, without any CPL script or context.

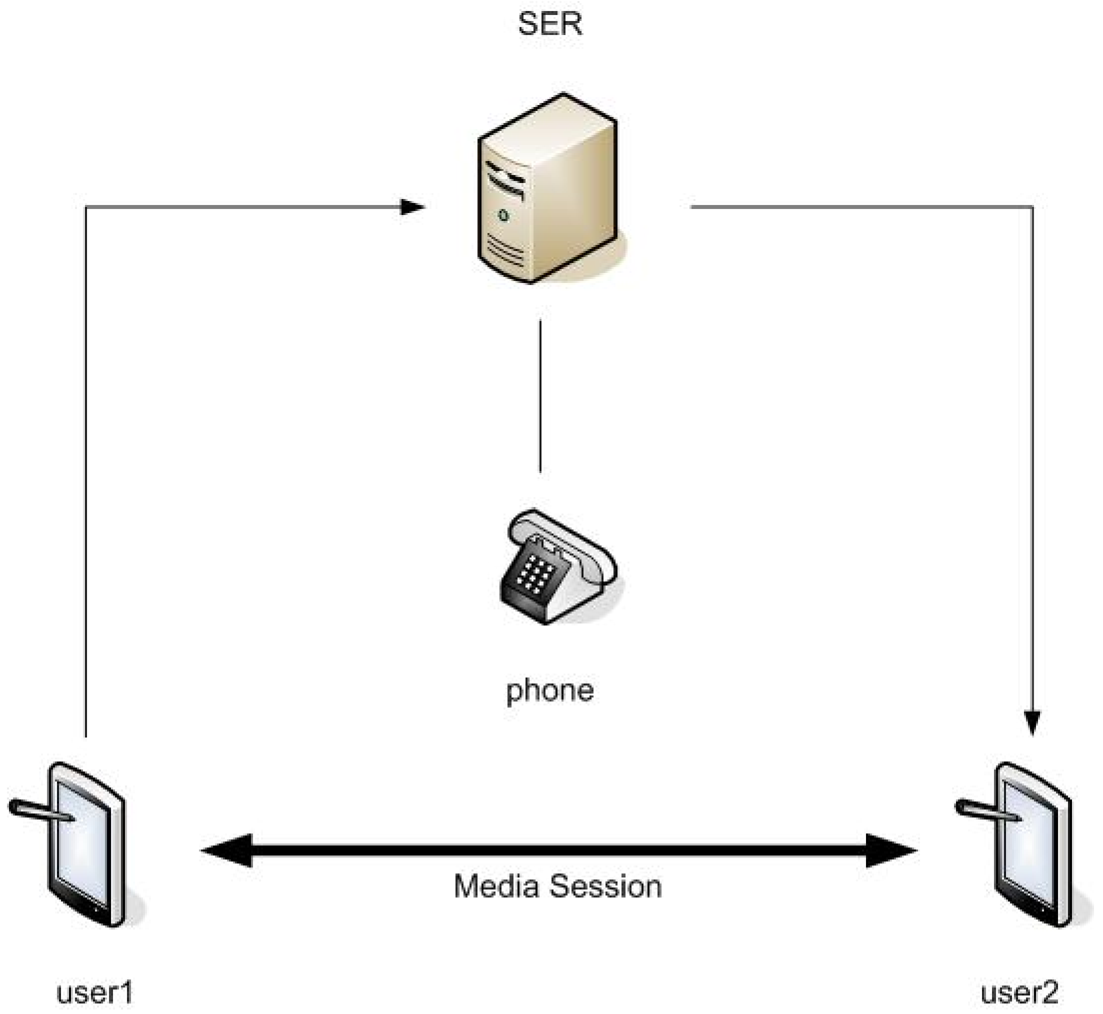
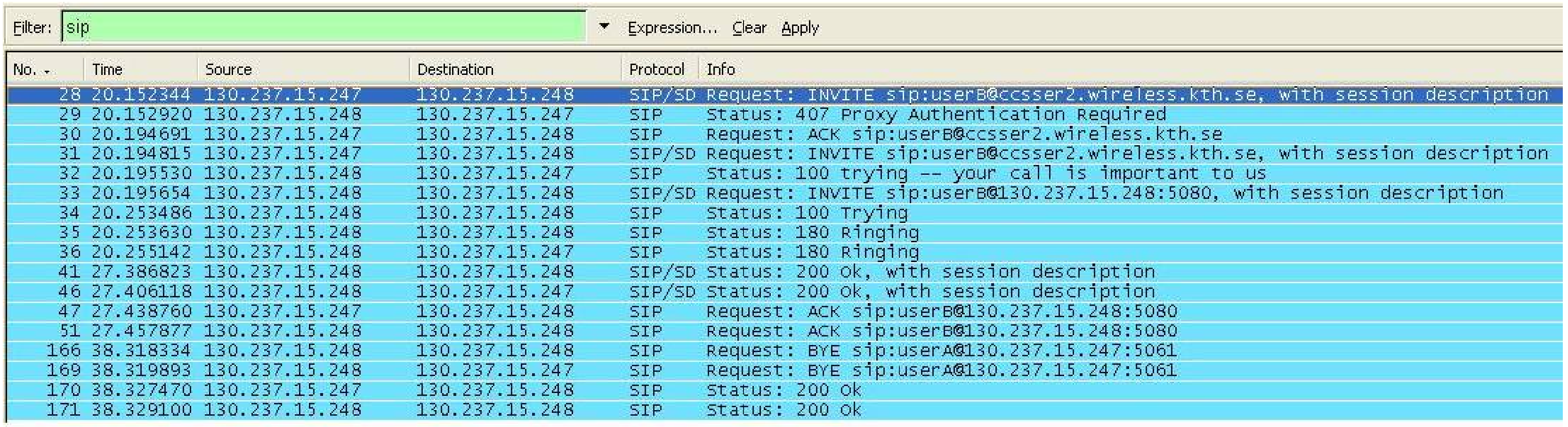


Figure 7: Architecture Test 1

Figure 7 represents the basic architecture of our test number 1. Figure 8 shows traffic captured using ethereal; it shows the SIP messages exchanged during the call from user1 to user2. Ethereal is a Network Protocol Analyzer to monitor network traffic (this software is now called Wireshark [41]). It is available on both Windows and Linux platforms.



```
Filter: sip                                  Expression...  Clear  Apply

No.  Time      Source          Destination     Protocol Info
 28 20.152344 130.237.15.247  130.237.15.248  SIP/SD  Request: INVITE sip:userB@ccsser2.wireless.kth.se, with session description
 29 20.152920 130.237.15.248  130.237.15.247  SIP     Status: 407 Proxy Authentication Required
 30 20.194691 130.237.15.247  130.237.15.248  SIP     Request: ACK sip:userB@ccsser2.wireless.kth.se
 31 20.194815 130.237.15.247  130.237.15.248  SIP/SD  Request: INVITE sip:userB@ccsser2.wireless.kth.se, with session description
 32 20.195530 130.237.15.248  130.237.15.247  SIP     Status: 100 trying -- your call is important to us
 33 20.195654 130.237.15.248  130.237.15.248  SIP/SD  Request: INVITE sip:userB@130.237.15.248:5080, with session description
 34 20.253486 130.237.15.248  130.237.15.248  SIP     Status: 100 Trying
 35 20.253630 130.237.15.248  130.237.15.248  SIP     Status: 180 Ringing
 36 20.255142 130.237.15.248  130.237.15.247  SIP     Status: 180 Ringing
 41 27.386823 130.237.15.248  130.237.15.248  SIP/SD  Status: 200 Ok, with session description
 46 27.406118 130.237.15.248  130.237.15.247  SIP/SD  Status: 200 Ok, with session description
 47 27.438760 130.237.15.247  130.237.15.248  SIP     Request: ACK sip:userB@130.237.15.248:5080
 51 27.457877 130.237.15.248  130.237.15.248  SIP     Request: ACK sip:userB@130.237.15.248:5080
166 38.318334 130.237.15.248  130.237.15.248  SIP     Request: BYE sip:userA@130.237.15.247:5061
169 38.319893 130.237.15.248  130.237.15.247  SIP     Request: BYE sip:userA@130.237.15.247:5061
170 38.327470 130.237.15.247  130.237.15.248  SIP     Status: 200 Ok
171 38.329100 130.237.15.248  130.237.15.248  SIP     Status: 200 Ok
```

Figure 8: Ethereal, Test 1

We can note that the time from when the UA of the user1 sends the SIP INVITE message to the Ringing message from the user2 is 0.101286 seconds.

We repeat the test 30 times, and we obtain the following result:

	INVITE	Ringing	Time	Diff of average
1	20,152344	20,253630	0,101286	0,034938
2	117,702719	117,809281	0,106562	0,029663
3	126,715461	126,833825	0,118364	0,017861
4	139,538351	139,651230	0,112879	0,023346
5	149,782708	149,893041	0,110333	0,025892
6	155,733709	155,871073	0,137364	-0,001139
7	161,127515	161,392235	0,264720	-0,128496
8	166,585686	166,712648	0,126962	0,009263
9	171,769874	171,886107	0,116233	0,019991
10	553,090685	553,322886	0,232201	-0,095977
11	558,892134	559,005932	0,113798	0,022427
12	3,951276	4,070015	0,118739	0,017486
13	10,735853	10,886687	0,150834	-0,014609
14	19,760831	19,898288	0,137457	-0,001233
15	27,248269	27,368158	0,119889	0,016335
16	34,074074	34,161597	0,087523	0,048702
17	41,191088	41,306025	0,114937	0,021288
18	48,734625	48,853088	0,118463	0,017762
19	54,967532	55,090404	0,122872	0,013352
20	4,350334	4,503874	0,153540	-0,017315
21	11,463968	11,579871	0,115903	0,020321
22	18,652198	18,868604	0,216406	-0,080182
23	25,389580	25,522704	0,133124	0,003100
24	32,142061	32,288779	0,146718	-0,010493
25	38,212478	38,328973	0,116495	0,019729
26	45,179082	45,270230	0,091148	0,045077
27	61,360179	61,482765	0,122586	0,013639
28	70,621772	70,850387	0,228615	-0,092391
29	3,720218	3,854924	0,134706	0,001519
30	13,740835	13,856913	0,116078	0,020147
		Average	0,136225	
		Variance	0,001837	

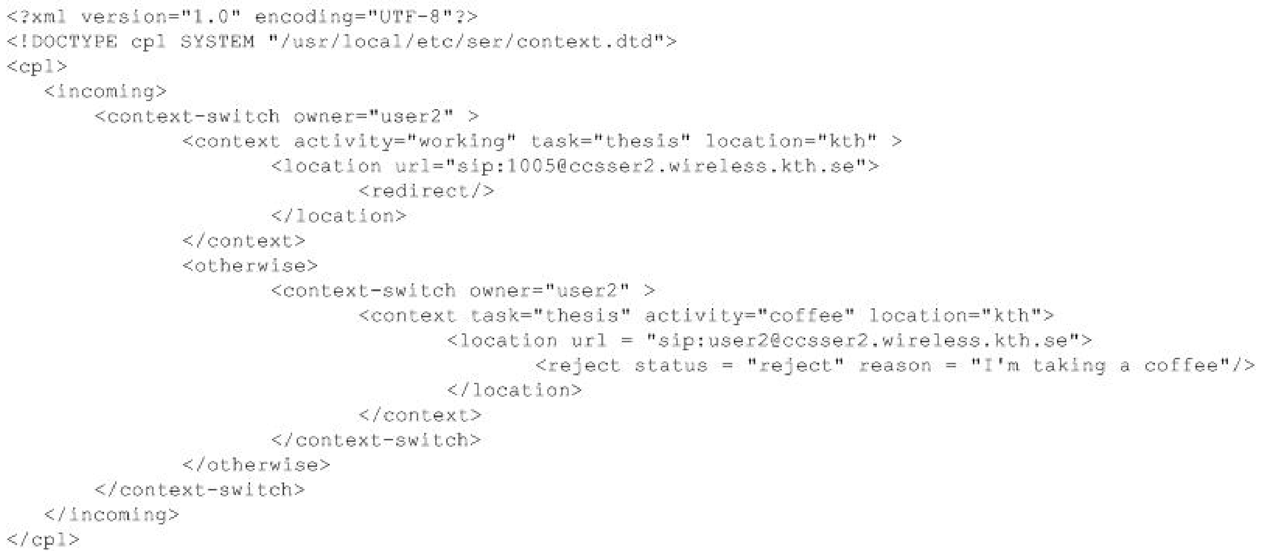
Table 2: Test 1 table results

All the values are expressed in seconds. We obtained all of these values in different captures, in other words, we didn't perform the test 30 times at the same time. Thus we can accommodate the fluctuations due to other traffic in the net.

The INVITE column represents the time that we send the INVITE message, the Ringing column indicated when we receive the Ringing message, and the Time column, gives the time that this process took. We calculated the average, 0.136225 seconds, and the Variance, 0.001837, from these 30 trials. We will compare these results with the time required for a response when using CPL scripts and context information.

First, we need to modify our SER source code. To do so, we simply follow the instructions in Devlic's report [42]. Once we have our SER working, we can continue with our tests.

For Test 2, Test 3, and Test 4 we upload the following CPL script to our SER for user2:



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cpl SYSTEM "/usr/local/etc/ser/context.dtd">
<cpl>
   <incoming>
       <context-switch owner="user2" >
              <context activity="working" task="thesis" location="kth" >
                     <location url="sip:1005@ccsser2.wireless.kth.se">
                            <redirect/>
                     </location>
              </context>
              <otherwise>
                     <context-switch owner="user2" >
                            <context task="thesis" activity="coffee" location="kth">
                                   <location url = "sip:user2@ccsser2.wireless.kth.se">
                                          <reject status = "reject" reason = "I'm taking a coffee"/>
                                   </location>
                            </context>
                     </context-switch>
              </otherwise>
       </context-switch>
   </incoming>
</cpl>
```

Figure 9: CPL script user2onLab

We upload it manually, with the command:

serctl fifo LOAD_CPL user@domain /path/to/cpl/script

Where *user* is user2 and *domain* is ccsser2.wireless.kth.se.

To upload the CPL script, we could also have used the CPLEd developed by Oukhay Younes and Alisa Devlic.

5.2 Test 2

The aim of this test is evaluate a call when a CPL script for the user2 has been uploaded to SER and it has a context defined. In this test we will use this context for the user2:

- task = “relax”
- activity = “nothing”
- location = “home”

Given this context we expect that user1 can establish a call with the user2, due to user2 not being busy and because there is no context specified in the CPL script that matches the given context. Notice that in the CPL script we only specify actions if we want to reject the call, proxy, redirect, …. In all other cases, the user will get the call.

The following figure represents the Test 2 architecture:

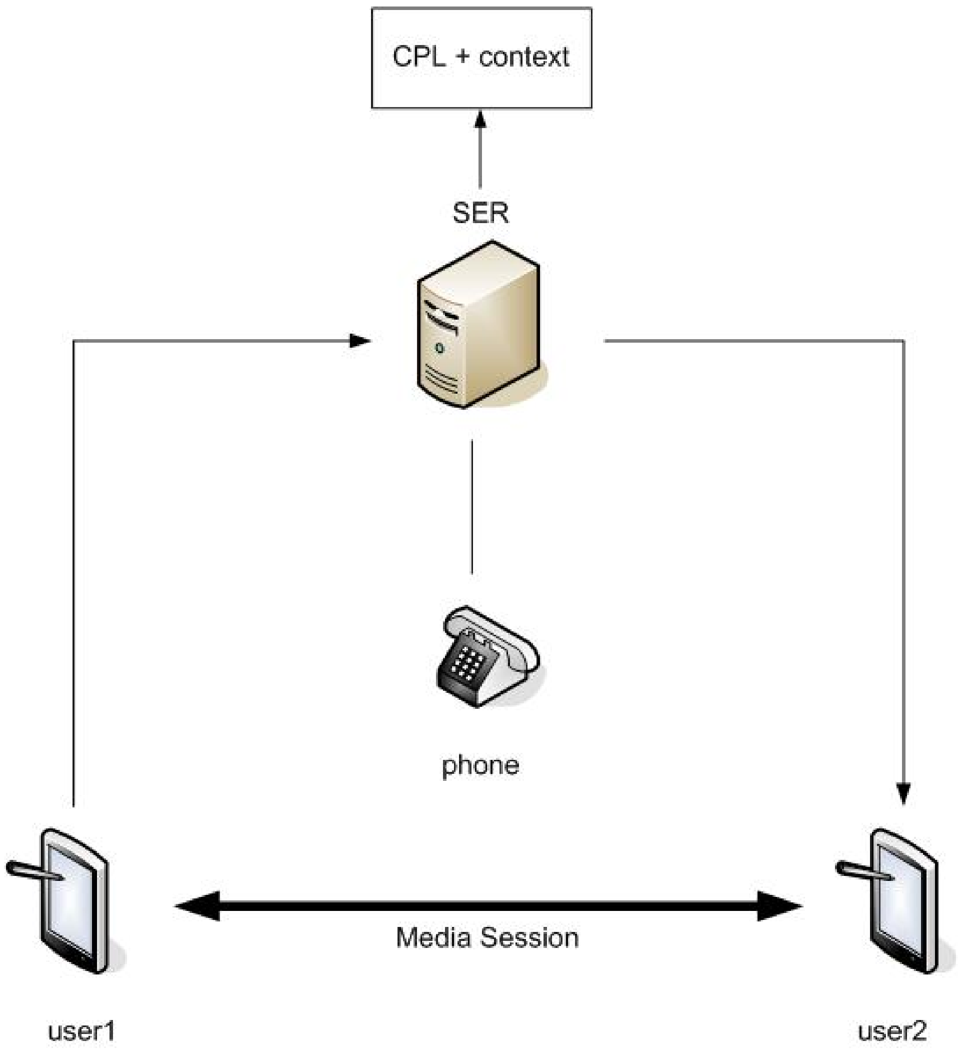


Figure 10: Architecture Test 2

To upload the context we use the following command inside the MySQL:

```
mysql> INSERT INTO context
-> (user, location, task, activity)
-> values
-> (“user2”, “home”, “relax”, “nothing”);
```

This uploads our context information into the context table in MySQL. Similar to the CPL script, we could also have used the program CPLEd.

However, during this test we found several problems that we want to mention. Initially we used the same ser.cfg file as Alisa Devlic used in her work, but in our case it did not work (perhaps we copied something wrong). So, when we try to call user2, the UA of user1 always displays the message: *Call failed: 500 CPL script exec failed*.

To solve this problem, we replaced the ser.cfg file with a new one, then all the problems disappeared, and we could continue with our test. For this test we used the ser.cfg that is shown Appendix 3. Repeating the same test as in Test 1 and we obtain the following results:

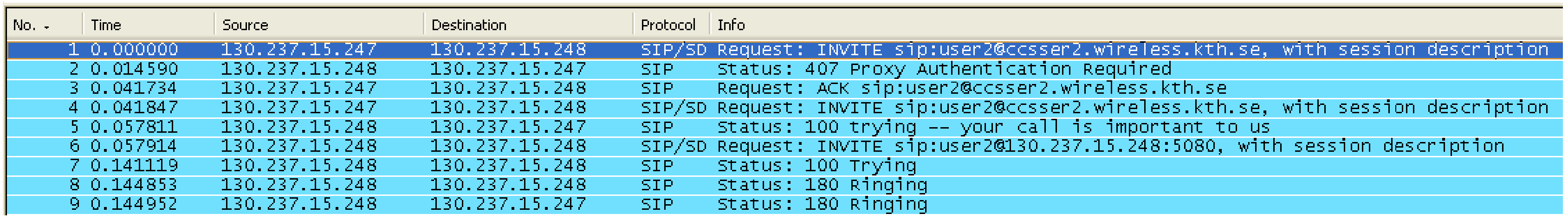
	INVITE	Ringing	Time	Diff of average
1	4,086841	4,244369	0,157528	0,039613
2	12,009106	12,147871	0,138765	0,058376
3	31,621946	31,985878	0,363932	-0,166791
4	50,622310	50,916659	0,294349	-0,097208
5	60,999938	61,132839	0,132901	0,064240
6	72,446276	72,590862	0,144586	0,052555
7	127,078874	127,267285	0,188411	0,008730
8	142,565023	142,798183	0,233160	-0,036019
9	191,938958	192,199043	0,260085	-0,062944
10	217,898031	218,245077	0,347046	-0,149905
11	3,432927	3,577780	0,144853	0,052288
12	11,223797	11,378221	0,154424	0,042717
13	19,363404	19,475647	0,112243	0,084898
14	33,284985	33,448364	0,163379	0,033762
15	43,805537	44,046220	0,240683	-0,043542
16	54,492476	54,640736	0,148260	0,048881
17	66,068379	66,288039	0,219660	-0,022519
18	76,163057	76,347641	0,184584	0,012557
19	85,364549	85,507101	0,142552	0,054589
20	97,701470	98,276637	0,575167	-0,378026
21	18,780874	18,954576	0,173702	0,023439
22	26,050387	26,116634	0,066247	0,130894
23	33,910705	34,086206	0,175501	0,021640
24	76,990356	77,144247	0,153891	0,043250
25	86,981729	87,169742	0,188013	0,009128
26	95,916013	96,021438	0,105425	0,091716
27	108,656719	108,872641	0,215922	-0,018781
28	121,556445	121,706783	0,150338	0,046803
29	167,156166	167,330496	0,174330	0,022811
30	187,353310	187,517596	0,164286	0,032855
		Average	0,197141	
		Variance	0,009426	

Table 3: Test 2 table results

The format and the method to get the values shown in the table are the same as in Test 1.

The resulting average is 0.197141 seconds, which is approximately 61ms more than in Test 1. One reason for this could be that SER now has to check if there is a match for this context (i.e., that specified in user2's CPL script). The variance was 0.009426.

The next figure represents one part of the Ethereal capture:



```
No.  Time      Source          Destination     Protocol Info
  1 0.000000  130.237.15.247  130.237.15.248  SIP/SD Request: INVITE sip:user2@ccsser2.wireless.kth.se, with session description
  2 0.014590  130.237.15.248  130.237.15.247  SIP    Status: 407 Proxy Authentication Required
  3 0.041734  130.237.15.247  130.237.15.248  SIP    Request: ACK sip:user2@ccsser2.wireless.kth.se
  4 0.041847  130.237.15.247  130.237.15.248  SIP/SD Request: INVITE sip:user2@ccsser2.wireless.kth.se, with session description
  5 0.057811  130.237.15.248  130.237.15.247  SIP    Status: 100 trying -- your call is important to us
  6 0.057914  130.237.15.248  130.237.15.248  SIP/SD Request: INVITE sip:user2@130.237.15.248:5080, with session description
  7 0.141119  130.237.15.248  130.237.15.248  SIP    Status: 100 Trying
  8 0.144853  130.237.15.248  130.237.15.248  SIP    Status: 180 Ringing
  9 0.144952  130.237.15.248  130.237.15.247  SIP    Status: 180 Ringing
```

Figure 11: Ethereal, Test 2

5.3 Test 3

In this test we repeat the same basic process as in test 2, but change the context and compare the results.

The new context is:

- task = “thesis”
- activity = “working”
- location = “kth”

Note that we have only changed the activity, the rest of parameters are the same. In our case, we use the same command as in the test 2, and we again upload all the parameters. As previously noted, in the future it would be interesting to upload only the changes; in this case only uploading the activity.

In this scenario user2 is working, so if there is an incoming call for him, it must be redirected to the Cisco phone, as defined in the CPL script. This is shown in figure 11:

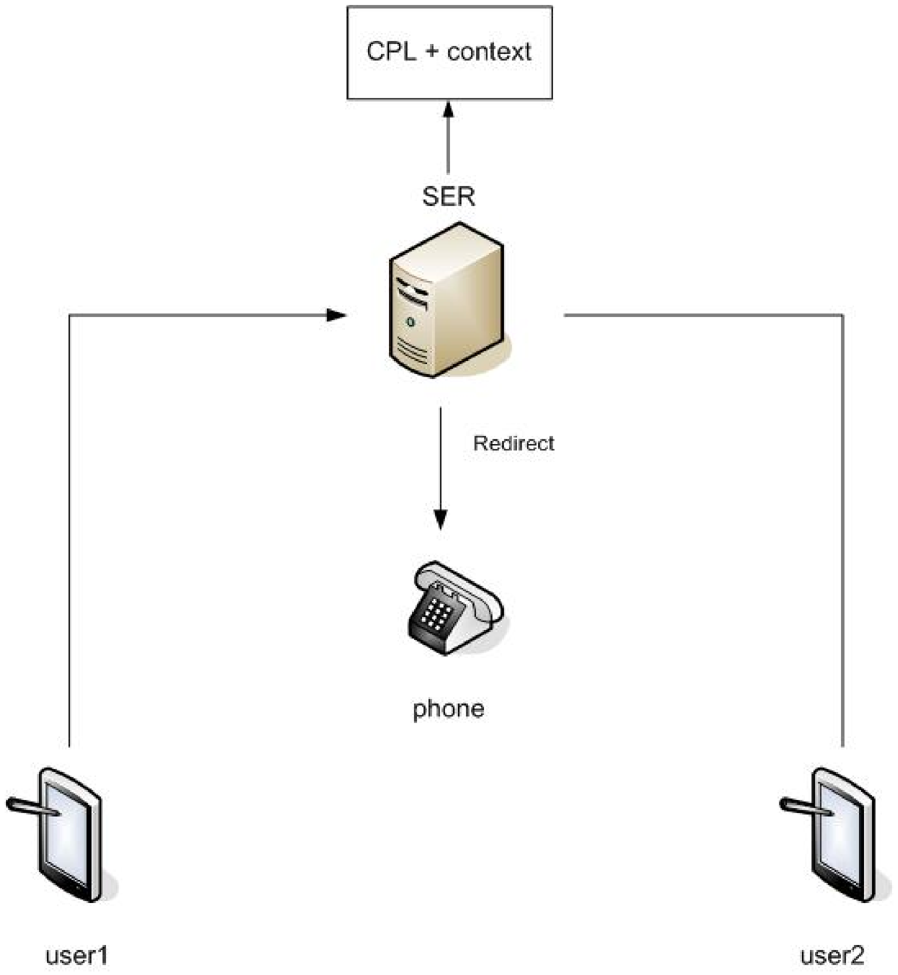
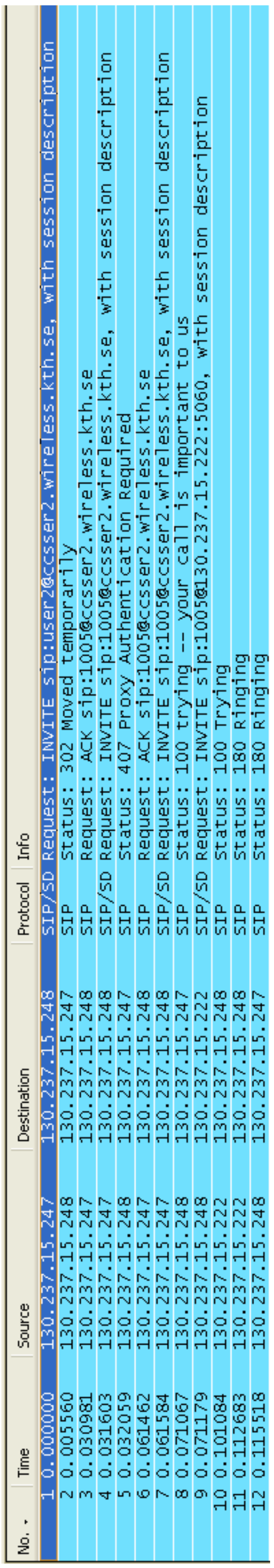


Figure 12: Architecture Test 3

A part of the capture of the Ethereal:



```
No. .  Time       Source          Destination     Protocol Info
     1 0.000000   130.237.15.247  130.237.15.248  SIP/SD Request: INVITE sip:user2@ccsser2.wireless.kth.se, with session description
     2 0.005560   130.237.15.248  130.237.15.247  SIP    Status: 302 Moved temporarily
     3 0.030981   130.237.15.247  130.237.15.248  SIP    Request: ACK sip:1005@ccsser2.wireless.kth.se
     4 0.031603   130.237.15.247  130.237.15.248  SIP/SD Request: INVITE sip:1005@ccsser2.wireless.kth.se, with session description
     5 0.032059   130.237.15.248  130.237.15.247  SIP    Status: 407 Proxy Authentication Required
     6 0.061462   130.237.15.247  130.237.15.248  SIP    Request: ACK sip:1005@ccsser2.wireless.kth.se
     7 0.061584   130.237.15.247  130.237.15.248  SIP/SD Request: INVITE sip:1005@ccsser2.wireless.kth.se, with session description
     8 0.071067   130.237.15.248  130.237.15.247  SIP    Status: 100 trying -- your call is important to us
     9 0.071179   130.237.15.248  130.237.15.222  SIP/SD Request: INVITE sip:1005@130.237.15.222:5060, with session description
    10 0.101084   130.237.15.222  130.237.15.248  SIP    Status: 100 Trying
    11 0.112683   130.237.15.222  130.237.15.248  SIP    Status: 180 Ringing
    12 0.115518   130.237.15.248  130.237.15.247  SIP    Status: 180 Ringing
```

Figure 13: Ethereal, Test 3

We repeat the same process as in Test 1 and Test 2 and we obtain these results:

	INVITE	Ringing	Time	Diff of average
1	6,334215	6,446898	0,112683	0,144343
2	12,487825	12,600524	0,112699	0,144327
3	20,082678	20,342775	0,260097	-0,003071
4	27,451789	27,572470	0,120681	0,136345
5	34,964446	35,252764	0,288318	-0,031292
6	43,578420	44,493881	0,915461	-0,658436
7	51,024491	51,248978	0,224487	0,032538
8	58,197369	58,318422	0,121053	0,135973
9	66,987221	67,238438	0,251217	0,005809
10	78,474812	78,593365	0,118553	0,138472
11	3,223346	3,382614	0,159268	0,097757
12	7,999525	8,125948	0,126423	0,130603
13	15,027224	15,169764	0,142540	0,114485
14	21,431182	21,662919	0,231737	0,025289
15	28,823031	28,957970	0,134939	0,122087
16	35,744421	35,930579	0,186158	0,070868
17	44,564402	44,854976	0,290574	-0,033549
18	51,941757	52,120896	0,179139	0,077887
19	58,515861	58,804096	0,288235	-0,031210
20	65,097334	66,009772	0,912438	-0,655412
21	3,581345	4,686174	1,104829	-0,847804
22	11,797563	11,912742	0,115179	0,141847
23	18,141712	18,267854	0,126142	0,130883
24	24,811243	24,947043	0,135800	0,121226
25	31,649043	31,857052	0,208009	0,049016
26	38,017910	38,146685	0,128775	0,128251
27	44,843338	44,970906	0,127568	0,129458
28	51,097261	51,209442	0,112181	0,144845
29	57,451123	57,789663	0,338540	-0,081514
30	64,272696	64,409738	0,137042	0,119983
		Average	0,257025	
		Variance	0,064824	

Table 4: Test 3 table results

The format of the table is the same as in Test 1 and Test 2.

Now we obtain an average of 0.257025 seconds, roughly 60ms longer than in the case of Test 2. One reason could be that now the call is redirected to the Cisco phone, so we need to send a new INVITE message. We can see that in the previous Ethereal capture. First user1 sends the SIP INVITE message to user2. The SER reply with the message: *302 Moved temporarily*, and with the new SIP address for the user1, 1005@ccsser2.wireless.kth.se. Then user2 sends the new INVITE message to that address.

5.4 Test 4

This test shows what happens when we uploaded a context which has a reject action in the CPL script. This is shown in figure 14:

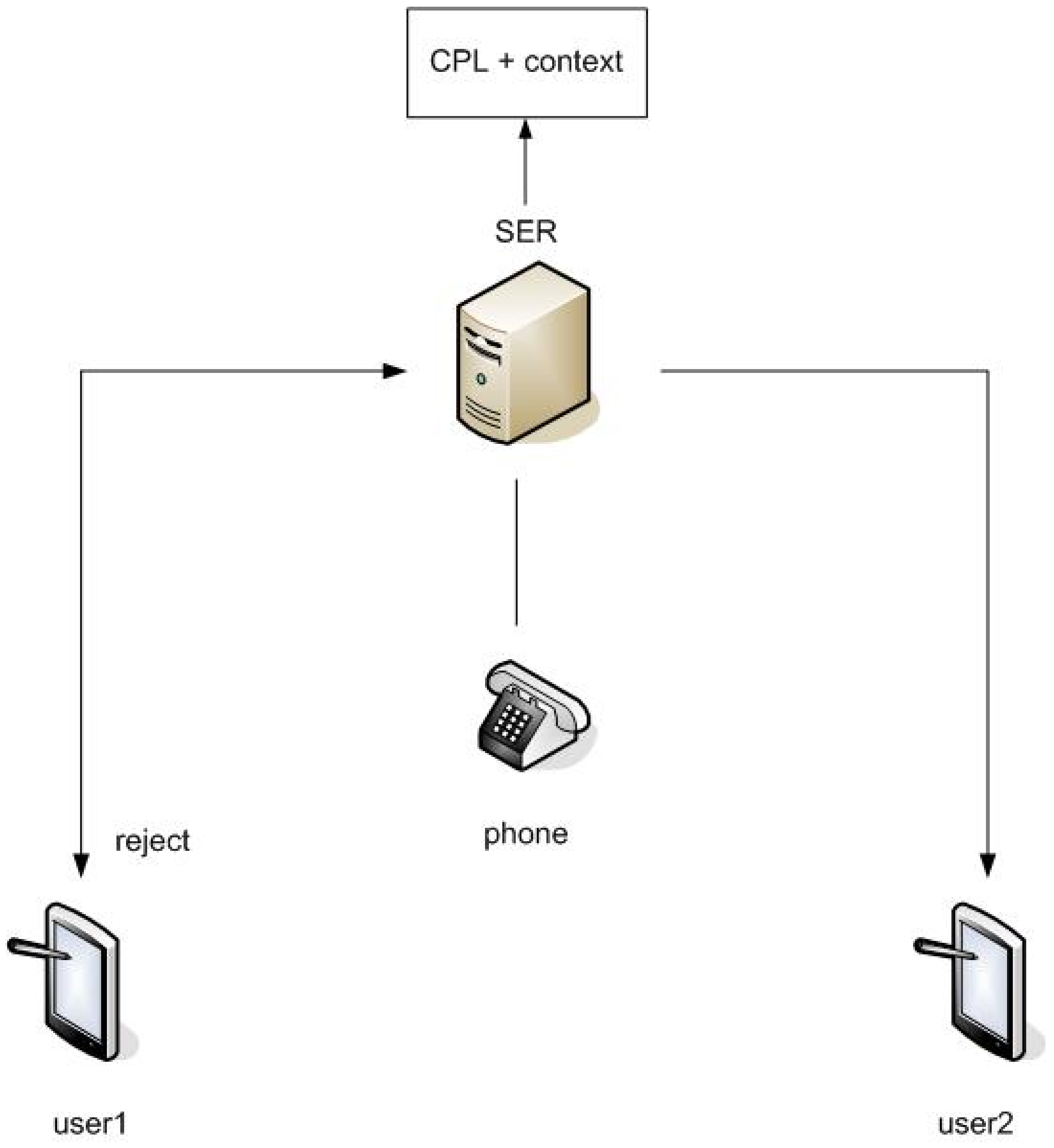
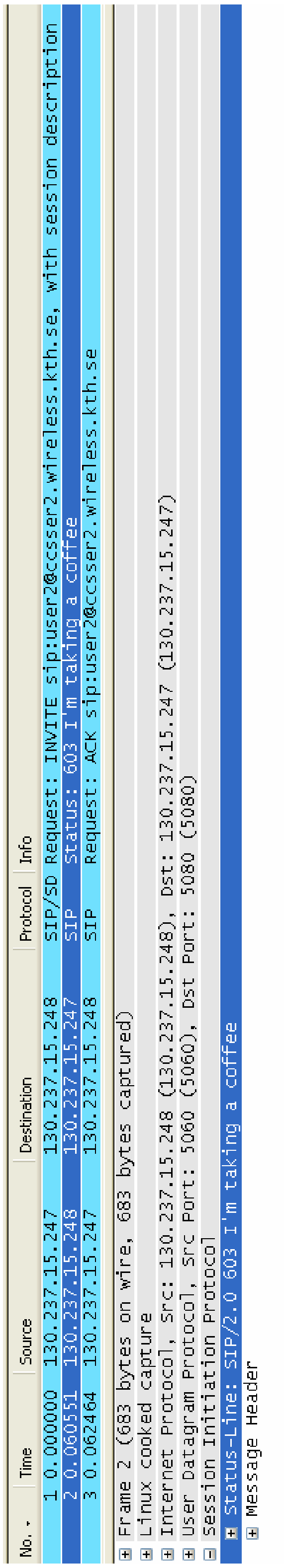


Figure 14: Architecture Test 4

Now, user2 has the following context:

- task = “thesis”
- activity = “coffee”
- location = “kth”

In the CPL script we defined that in this context the action must be to reject the call and send the message: “I’m taking a coffee”. We can see what happens in the capture of the Ethereal:



```
No. . Time      Source          Destination     Protocol Info
    1 0.000000  130.237.15.247  130.237.15.248  SIP/SD Request: INVITE sip:user2@ccsser2.wireless.kth.se, with session description
    2 0.060551  130.237.15.248  130.237.15.247  SIP    Status: 603 I'm taking a coffee
    3 0.062464  130.237.15.247  130.237.15.248  SIP    Request: ACK sip:user2@ccsser2.wireless.kth.se

⊞ Frame 2 (683 bytes on wire, 683 bytes captured)
⊞ Linux cooked capture
⊞ Internet Protocol, Src: 130.237.15.248 (130.237.15.248), Dst: 130.237.15.247 (130.237.15.247)
⊞ User Datagram Protocol, Src Port: 5060 (5060), Dst Port: 5080 (5080)
⊟ Session Initiation Protocol
  ⊞ Status-Line: SIP/2.0 603 I'm taking a coffee
  ⊞ Message Header
```

Figure 15: Ethereal, Test 4

The UA of the user1 receives the message: “I’m taking a coffee”, thus the SER reject the call correctly.

In this test we didn’t report any timings because there is no Ringing message, but it is useful to show another possibility of using SER with CPL and context.

So, in conclusion, in these tests we saw the differences between when the SER process a call with or without a CPL script and context. Also, we compare different contexts defined for the user.

To understand the process we utilize the time difference between the first SIP INVITE message and the first SIP Ringing message. We notice that we spend more time when we redirect the call, as this invokes additional logic. If we have a CPL script and context for the user that is not defined in the CPL, we spend more time than if we don’t have CPL and context. Given the simple CPL scrip and context, it is not clear how this will scale with more complex scripts, however, Alisa Devlic’s earlier report [9] gives some insight into this scaling.

6. Conclusions

In this thesis project, we tried to combine some of the new technologies of voice over IP (VoIP) with context awareness services for mobile users to create a demonstrator for a typical routine of a student in Kista. First we had to learn and understand all the existing technology that can be used to develop our idea. We learned about SIP, the SIP express router, the call processing language, the web ontology language, etc. All of these are really new technologies and it was important to know and understand it correctly to use them in the right way. At the same time, we built upon the previous work of other students, without this work, this thesis would have been impossible.

In order to create a demonstrator for a typical routine of a student in Kista, the best and the most logical way was to define the user's routine. Here we have examined the routine of a student that was doing his masters thesis project and also taking a course.

Once we extracted his routine we could create CPL scripts for each state and the OWL scripts for each context. Here we used the prior work of Oukhay Younes and Alisa Devlic.

Next, we had to understand the operation of our SIP Express Router, SER. In the lab we had one computer with the SER that was used and "created" by Oukhay and Alisa. But we found some problems with it and we installed a new version of SER in another computer. First we found some problems with the configuration, but finally got it to work. Next we practiced with this SER and the SIP user agents.

Next we tried to extend this "basic" SER. We changed the configuration and tried to make the CPL extensions work.

In the prior work of Oukhay and Alisa, they used an application that manually uploaded the CPL and OWL scripts to the SER. However, this application was written specifically for that computer, and when we tried to port it to our computer we found several problems and were unable to make it work.

We should mention the parallel theses by Athanasios Karapantelakis [43] who is porting miniSIP [44] to the iPAQ. This port would be a very useful as part of a real demonstrator, as it would allow a mobile use to have SIP calls while on the move.

While this thesis project was a very interesting experience and I learn a lot of previously unknown technologies; I made some mistakes in my programming my work and could not obtain the final result that I hope for at the beginning of my thesis (i.e., a functional demonstrator). However, I can say that I'm satisfy with my work and that I learn a lot in the course of attempting to reach this goal.

In the next section we try to define the next steps and future work for some one who would like to continue this work.

6.1 Future work

This thesis provides a good starting point for future work. Some of the most obvious future work is described below:

miniSIP for the iPAQ

We need a mobile device to test our work. A good solution would be an iPAQ with miniSIP. This is the current work of Athanasios Karapantelakis. It would be useful to port the CPLEd application to it, in order to upload the OWL scripts manually from the mobile device. The next step would be an automatic upload application or a means to conditionalize the scripts.

Location

As we said earlier, it is important to know were the user is for our demonstrator. The work of Andreas Friberg [33], Lin Ji [34], and Behdis Zandieh [35] should be used to create an application for determine the user's location as input to the context information of this thesis. Such an application could advise the user to move to a new location that is better for their current task.

Groups of people

It would be very useful to our system if it could automatically compute the membership of different groups based on the contacts of the user. For example, this would enable the system to automatically distinguish if a call comes from a family member, friend, colleague, etc. This avoids the user having to specify the same rules again and again for different people. A good starting point might be The Friend of a Friend (FOAF) project [45]. Using the FOAF ontology language, we can create different groups that can then be treating differently based upon the user's preferences. This would make our system more intelligent and easier to configure.

Feedback application

We can create a feedback application that lets us to know if when we reject a call due to the context we are taking the right action. We can ask the user if the choice we made was ok or not, and using that information we can improve our subsequent decisions. This is important because we want to help the user, rather than complicate his or her life.

References

[1] Compact Oxford English Dictionary,
http://www.askoxford.com/concise_oed/context?view=uk, 2006

[2] A.K. Dey, G.D. Abowd, and D. Salber.
A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction,* v 16, n 2-4, 2001, p 97-166

[3] H.I. Myrhaug, N. Whitehead, A. Goker, T.E. Faegri, and T.C. Lech.
AmbieSense - a system and reference architecture for personalised context-sensitive information services for mobile users. *Ambient Intelligence. Second European Symposium, EUSAI 2004. Proceedings (Lecture Notes in Computer Science Vol.3295),* 2004, p 327-38

[4] Mari Korkea-aho, Department of Computer Science, Helsinki University of Technology, section 3: Existing Context-Aware Applications,
http://users.tkk.fi/~mkorkeaa/doc/context-aware.html, last access January 2007

[5] Human Dynamics Group at the MIT Media Lab Reality Mining project, Principal Investigator Nathan Eagle, http://reality.media.mit.edu/dataset.php, 2006

[6] “3 User Modeling: Identifying Structure in Routine”, section: “3.2 Models to Identify Location and Activity”, of N. Eagle and A. Pentland (2005), “Reality Mining: Sensing Complex Social Systems”, Personal and Ubiquitous Computing, Vol 10, #4, 2006. http://reality.media.mit.edu/pdfs/realitymining.pdf

[7] Oukhay Younes, Context Aware Services, Master thesis, School of Information and Communication Technology (ICT), Royal Institute of Technology (KTH), January 25th 2006.

[8] Alisa Devlic, Semantic agents for location-aware service provisioning in mobile network, Master thesis, Faculty of Electrical Engineering and Computing, University of Zagreb, 2005

[9] Alisa Devlic, CPL extensions, Report for course 2G1325/2G5564, Royal Institute of Technology (KTH), 31 May 2006

[10] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg. “SIP: Session Initiation Protocol”, RFC2545, ftp://ftp.rfc-editor.org/in-notes/rfc2543.txt, March 1999

[11] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler. “SIP: Session Initiation Protocol” RFC3261, ftp://ftp.rfc-editor.org/in-notes/rfc3261.txt, June 2002

[12] T. Dierks, and C. Allen, "The TLS Protocol Version 1.0", Internet RFC 2246, January

1999.

[13] Netscape Communication Corp, “The SSL protocol”,
http://developer.netscape.com/docs/manuals/security/sslin/contents.htm#104
1986, September 1998

[14] Gerald Q. "Chip" Maguire Jr., Lecture 2G1325/2G5564 Practical Voice Over IP (VoIP): SIP and related protocols Spring 2006, Period 4., KTH Information and Communication Technology, Spring 2006, Period 4

[15] Session Description Protocol, SDP, Wikipedia
http://en.wikipedia.org/wiki/Session_Description_Protocol, last access January 2007

[16] Johannes Jansson, Context-aware Service Allocation in Personal Area Networks, School of Information and Communication Technology (ICT), Royal Institute of Technology (KTH), 2004

[17] GNU General Public License, Version 2, June 1991
http://www.gnu.org/licenses/gpl.txt

[18] iptel.org, SIP Express Router, http://www.iptel.org/ser, last access January 2007

[19] OpenSER - the Open Source SIP Server, http://www.openser.org/, last acces January 2007

[20] Paul Hazlett, Simon Miles, Greger V.Teigre, SER-Getting Started, 12. Appendix The Call Processing Language (CPL), Last access January 2007

[21] Extensible Markup Language (XML), http://www.w3.org/XML/, last access January 2007

[22] The World Wide Web Consortium (W3C), http://www.w3.org/, last access January 2007

[23] Standard Generalized Markup Language (SGML),
http://www.w3.org/MarkUp/SGML/, last access January 2007

[24] XML stands for eXtensible Markup Language, http://www.w3schools.com/xml/, last access January 2007

[25] J. Lennox, X. Wu, H. Schulzrinne; “Call Processing Language (CPL)” ftp://ftp.rfc-editor.org/in-notes/rfc3880.txt, October 2004

[26] H.323, Wikipedia, http://en.wikipedia.org/wiki/H.323, last access January 2007

[27] Grigoris Antoniou and Frank van Harmelen. A Semantic Web Primer. The MIT Press, Cambridge, Massachusetts, 2004.

[28] Resource Description Framework (RDF), http://www.w3.org/RDF/, last access January 2007

[29] OWL Web Ontology Language, W3C Recommendation 10 February 2004 http://www.w3.org/TR/owl-features/, last access January 2007

[30] Semantic Web project, http://www.w3.org/2001/sw/, last access January 2007

[31] Figure 3: A Semantic Web stack from Tim Berners-Lee, source of inspiration: http://www.dajobe.org/talks/copenhagen-dc/slide8.html, last access January 2007

[32] DAML-ONT Initial Release. http://www.daml.org/2000/10/daml-ont.html, last access January 2007

[33] Andreas Friberg, Increasing Location Precision Using WLAN Signals, Master thesis, Royal Institute of Technology, in preparation

[34] Lin Ji, Increasing Accuracy of Location Determination: Exploiting Phase Change Reconstruction and Timing Measurements, Master thesis, Royal Institute of Technology, in preparation

[35] Behdis Zandieh, Building 3D model of indoor wireless LANs, Master thesis, Royal Institute of Technology, in preparation

[36] OWL-S 1.0 Release, http://www.daml.org/services/owl-s/1.0/, last access January 2007

[37] Oukhay Younes, Context Aware Services, Appendix G: CPLEd added classes, Master thesis, School of Information and Communication Technology (ICT), Royal Institute of Technology (KTH), January 25th 2006.

[38] Alisa Devlic, CPL extensions, chapter 4. Context-aware VoIP prototype, Report for course 2G1325/2G5564, Royal Institute of Technology (KTH), 31 May 2006

[39] X-Lite, http://www.xten.com/, last access January 2007

[40] CISCO phone 7960 series, http://www.cisco.com/en/US/products/hw/phones/ps379/ps1855/index.html, last access January 2007

[41] Wireshark, http://www.wireshark.org/, last access January 2007

[42] Alisa Devlic, CPL extensions, section 4.1.4. CPL-C module extensions, Report for course 2G1325/2G5564, Royal Institute of Technology (KTH), 31 May 2006

[43] Athanasios Karapantelakis, A mobile SIP client: From the user interface design to experimental evaluation of synchronized playout from multiple SIP User Agents, Master

thesis, Royal Institute of Technology, in preparation

[44] miniSIP, http://www.minisip.org/, last access January 2007

[45] The Friend of a Friend (FOAF) project, http://www.foaf-project.org/, last access January 2007

Appendix A

CPL Scripts

SergiOnHome2105_0830

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cpl SYSTEM "/home/devlic/From_Younes/CPLEd/CPLEd/CPLscripts/context.dtd">
<cpl>
  <subaction id="voicemail" >
    <location url="sip:sergi@voicemail.it.kth.se" />
  </subaction>
  <incoming>
    <time-switch>
      <time freq="weekly" duration="PT11H35M" dtstart="20060901T210500" byday="MO, TU, WE, TH,
FR" >
        <context-switch owner="sergi" >
          <context activity="nothing" task="rest" location="home" >
            <location url="sip:sergi@it.kth.se" >
              <proxy/>
            </location>
          </context>
          <otherwise>
            <sub ref="voicemail" />
          </otherwise>
        </context-switch>
      </time>
    </time-switch>
  </incoming>
</cpl>
```

SergiOnTravel0830_0920

```
<?xml version ="1.0" encoding ="UTF-8"?>
<!DOCTYPE cpl SYSTEM "/home/devlic/From_Younes/CPLEd/CPLEd/CPLscripts/context.dtd">
<cpl>
  <subaction id="voicemail" >
    <location url="sip:sergi@voicemail.it.kth.se" />
  </subaction>
  <incoming>
    <time-switch>
      <time freq="weekly" duration="PT0H50M" dtstart="20060901T083000" byday="MO, TU, WE, TH, FR"
>
        <context-switch owner="sergi" >
          <context activity="walking" task="travel" location="kth" >
            <location url="sip:sergi@it.kth.se" />
          </context>
          <context activity="walking" task="travel" location="unknown" >
            <location url="tel:+34649154318" >
              <redirect/>
            </location>
          </context>
          <otherwise>
            <sub ref="voicemail" />
          </otherwise>
        </context-switch>
      </time>
    </time-switch>
  </incoming>
</cpl>
```

SergiOnLab0920_1300

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cpl SYSTEM "/home/devlic/From_Younes/CPLEd/CPLEd/CPLscripts/context.dtd">
<cpl>
  <subaction id="voicemail" >
    <location url="sip:sergi@voicemail.it.kth.se" />
  </subaction>
  <incoming>
    <time-switch>
      <time freq="weekly" duration="PT4H20" dtstart="20060901T092000" byday="MO, TU, WE, TH, FR" >
        <context-switch owner="sergi" >
          <context activity="working" task="thesis" location="kth" >
            <priority-switch>
              <priority equal="emergency" >
                <location url="sip:sergi@it.kth.se" >
                  <proxy/>
                </location>
              </priority>
            </priority-switch>
            <address-switch field="owner" >
              <address subdomain-of="family.com" >
                <location url="sip:sergi@it.kth.se" >
                  <proxy/>
                </location>
              </address>
            </address-switch>
            <otherwise>
              <sub ref="voicemail" />
            </otherwise>
          </context>
          <context activity="coffee" task="thesis" location="kth" >
            <priority-switch>
              <priority equal="emergency" >
                <location url="sip:sergi@it.kth.se" >
                  <proxy/>
                </location>
              </priority>
              <address-switch field="owner" >
                <address subdomain-of="family.com" >
                  <location url="sip:sergi@it.kth.se" >
                    <proxy/>
                  </location>
                </address>
                <address subdomain-of="friends.com" >
                  <location url="sip:sergi@it.kth.se" >
          <proxy/>
         </location>
        </address>
       </address-switch>
       <otherwise>
        <sub ref="voicemail" />
       </otherwise>
      </priority-switch>
     </context>
     <context activity="wc" task="thesis" location="kth" >
      <priority-switch>
       <priority equal="emergency" >
        <location url="sip:sergi@it.kth.se" >
         <proxy/>
        </location>
       </priority>
       <otherwise>
        <sub ref="voicemail" />
       </otherwise>
      </priority-switch>
     </context>
     <otherwise>
      <sub ref="voicemail" />
     </otherwise>
    </context-switch>
   </time>
  </time-switch>
 </incoming>
</cpl>
```

SergiOnTravel1300_1315

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE cpl SYSTEM "/home/devlic/From_Younes/CPLEd/CPLEd/CPLscripts/context.dtd">
<cpl>
 <subaction id="voicemail" >
  <location url="sip:sergi@voicemail.it.kth.se" />
 </subaction>
 <incoming>
  <time-switch>
   <time freq="weekly" duration="PT0H15M" dtstart="20060901T130000" byday="MO, TU, WE, TH, FR"
>
    <context-switch owner="sergi" >
     <context activity="walking" task="travel" location="unknown" >
      <location url="tel:+34659154318" >
       <redirect/>
      </location>
     </context>
     <context activity="walking" task="travel" location="kth" >
      <location url="sip:sergi@it.kth.se" >
       <proxy/>
      </location>
     </context>
     <otherwise>
```

```
          <sub ref="voicemail" />
        </otherwise>
      </context-switch>
    </time>
  </time-switch>
 </incoming>
</cpl>
```

SergiOnLunch1315_1415

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cpl SYSTEM "/home/devlic/From_Younes/CPLEd/CPLEd/CPLscripts/context.dtd">
<cpl>
 <subaction id="voicemail" >
   <location url="sip:sergi@voicemail.it.kth.se" />
 </subaction>
 <incoming>
   <time-switch>
     <time freq="weekly" duration="PT1H" dtstart="20060901T131500" byday="MO, TU, WE, TH, FR" >
       <context-switch owner="sergi" >
         <context activity="lunching" task="lunch" location="kth" >
           <location url="sip:sergi@it.kth.se" >
             <proxy/>
           </location>
         </context>
         <otherwise>
           <sub ref="voicemail" />
         </otherwise>
       </context-switch>
     </time>
   </time-switch>
 </incoming>
</cpl>
```

SergiOnTravel1415_1430

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cpl SYSTEM "/home/devlic/From_Younes/CPLEd/CPLEd/CPLscripts/context.dtd">
<cpl>
 <subaction id="voicemail" >
   <location url="sip:sergi@voicemail.it.kth.se" />
 </subaction>
 <incoming>
   <time-switch>
     <time freq="weekly" duration="PT0H15M" dtstart="20060901T141500" byday="MO, TU, WE, TH, FR"
>
       <context-switch owner="sergi" >
         <context activity="walking" task="travel" location="unknown" >
           <location url="tel:+34659154318" >
             <redirect/>
           </location>
         </context>
```

```
          <context activity="walking" task="travel" location="kth" >
            <location url="sip:sergi@it.kth.se" >
              <proxy/>
            </location>
          </context>
          <otherwise>
            <sub ref="voicemail" />
          </otherwise>
        </context-switch>
      </time>
    </time-switch>
  </incoming>
</cpl>
```

SergiOnLab1430_1645

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cpl SYSTEM "/home/devlic/From_Younes/CPLEd/CPLEd/CPLscripts/context.dtd">
<cpl>
  <subaction id="voicemail" >
    <location url="sip:sergi@voicemail.it.kth.se" />
  </subaction>
  <incoming>
    <time-switch>
      <time freq="weekly" duration="PT2H15M" dtstart="20060901T143000" byday="MO, TU, WE, TH, FR"
>
        <context-switch owner="sergi" >
          <context activity="working" task="thesis" location="kth" >
            <priority-switch>
              <priority equal="emergency" >
                <location url="sip:sergi@it.kth.se" >
                  <proxy/>
                </location>
              </priority>
            </priority-switch>
            <address-switch field="owner" >
              <address subdomain-of="family.com" >
                <location url="sip:sergi@it.kth.se" >
                  <proxy/>
                </location>
              </address>
            </address-switch>
            <otherwise>
              <sub ref="voicemail" />
            </otherwise>
          </context>
          <context activity="coffee" task="thesis" location="kth" >
            <priority-switch>
              <priority equal="emergency" >
                <location url="sip:sergi@it.kth.se" >
                  <proxy/>
                </location>
              </priority>
              <address-switch field="owner" >
                <address subdomain-of="family.com" >
                  <location url="sip:sergi@it.kth.se" >
                    <proxy/>
                  </location>
                </address>
                <address subdomain-of="friends.com" >
                  <location url="sip:sergi@it.kth.se" >
                    <proxy/>
                  </location>
                </address>
              </address-switch>
              <otherwise>
                <sub ref="voicemail" />
              </otherwise>
            </priority-switch>
          </context>
          <context activity="wc" task="thesis" location="kth" >
            <priority-switch>
              <priority equal="emergency" >
                <location url="sip:sergi@it.kth.se" >
                  <proxy/>
                </location>
              </priority>
              <otherwise>
                <sub ref="voicemail" />
              </otherwise>
            </priority-switch>
          </context>
          <otherwise>
            <sub ref="voicemail" />
          </otherwise>
        </context-switch>
      </time>
    </time-switch>
  </incoming>
</cpl>
```

SergiOnTravel1645_1700

```
<?xml version="1.0" encoding="UTF-8"?>
<cpl>
 <subaction id="voicemail" >
  <location url="sip:sergi@voicemail.it.kth.se" />
 </subaction>
 <incoming>
  <time-switch>
   <time freq="weekly" duration="PT0H15M" dtstart="20060901T164500" byday="MO, TU, WE, TH, FR"
>
    <context-switch owner="sergi" >
      <context activity="walking" task="travel" location="unknown" >
        <location url="tel:+34659154318" >
          <redirect/>
        </location>
      </context>
```

```
          <context activity="walking" task="travel" location="kth" >
            <location url="sip:sergi@it.kth.se" >
              <proxy/>
            </location>
          </context>
          <otherwise>
            <sub ref="voicemail" />
          </otherwise>
        </context-switch>
      </time>
    </time-switch>
  </incoming>
</cpl>
```

SergiOnClass1700_2015

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cpl SYSTEM "/home/devlic/From_Younes/CPLEd/CPLEd/CPLscripts/context.dtd">
<cpl>
  <subaction id="voicemail" >
    <location url="sip:sergi@voicemail.it.kth.se" />
  </subaction>
  <incoming>
    <time-switch>
      <time freq="weekly" duration="PT3H15M" dtstart="20060901T170000" byday="MO, TU, WE, TH, FR"
>
        <context-switch owner="sergi" >
          <context activity="listening" task="class" location="kth" >
            <priority-switch>
              <priority equal="emergency" >
                <location url="sip:sergi@it.kth.se" >
                  <proxy/>
                </location>
              </priority>
              <otherwise>
                <sub ref="voicemail" />
              </otherwise>
            </priority-switch>
          </context>
          <context activity="break" task="class" location="kth" >
            <location url="ip:sergi@it.kth.se" >
              <proxy/>
            </location>
          </context>
          <otherwise>
            <sub ref="voicemail" />
          </otherwise>
        </context-switch>
      </time>
    </time-switch>
  </incoming>
</cpl>
```

SergiOnTravel2015_2105

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cpl SYSTEM "/home/devlic/From_Younes/CPLEd/CPLEd/CPLscripts/context.dtd">
<cpl>
  <subaction id="voicemail" >
    <location url="sip:sergi@voicemail.it.kth.se" />
  </subaction>
  <incoming>
    <time-switch>
      <time freq="weekly" duration="PT0H50M" dtstart="20060901T201500" byday="MO, TU, WE, TH, FR"
>
        <context-switch owner="sergi" >
          <context activity="walking" task="travel" location="unknown" >
            <location url="tel:+34659154318" >
              <redirect/>
            </location>
          </context>
          <context activity="walking" task="travel" location="kth" >
            <location url="sip:sergi@it.kth.se" >
              <proxy/>
            </location>
          </context>
          <otherwise>
            <sub ref="voicemail" />
          </otherwise>
        </context-switch>
      </time>
    </time-switch>
  </incoming>
</cpl>
```

Appendix B

OWL scripts:

SergiHome.owl

```
<?xml version="1.0"?>
<rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
        xmlns:acas="http://localhost/OWL/userContextOntology.owl#">

            <owl:Ontology rdf:about=""/>

            <acas:User rdf:ID="sergi">
                        <acas:hasContext>
                                    <acas:UserContext rdf:ID="sergiContext">
                                                <acas:hasOwner
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">sergi</acas:hasOwner>
                                                <acas:hasLocation
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">home</acas:hasLocation>
                                                <acas:hasTask
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">rest</acas:hasTask>
                                                <acas:hasActivity
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">nothing</acas:hasActivity>
                                    </acas:UserContext>
                        </acas:hasContext>
                        <acas:hasAddress>
                                    <acas:UserAddress rdf:ID="sergiAddress">
                                                <acas:hasHostName
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">ccsser1.wireless.kth.se</acas:hasHostName
>
                                                <acas:hasHostDomain
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">wireless.kth.se</acas:hasHostDomain>
                                                <acas:hasPhoneNumber
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">tel:+999999999</acas:hasPhoneNumber>
                                                <acas:hasPhoneDomain
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">1</acas:hasPhoneDomain>
                                    </acas:UserAddress>
                        </acas:hasAddress>
            </acas:User>
</rdf:RDF>
```

SergiTravelUnknown.owl

```
<?xml version="1.0"?>
<rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
        xmlns:acas="http://localhost/OWL/userContextOntology.owl#">

            <owl:Ontology rdf:about=""/>
```

```
		<acas:User rdf:ID="sergi">
			<acas:hasContext>
				<acas:UserContext rdf:ID="sergiContext">
					<acas:hasOwner
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">sergi</acas:hasOwner>
					<acas:hasLocation
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">unknown</acas:hasLocation>
					<acas:hasTask
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">travel</acas:hasTask>
					<acas:hasActivity
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">walking</acas:hasActivity>
				</acas:UserContext>
			</acas:hasContext>
			<acas:hasAddress>
				<acas:UserAddress rdf:ID="sergiAddress">
					<acas:hasHostName
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">ccsser1.wireless.kth.se</acas:hasHostName
>
					<acas:hasHostDomain
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">wireless.kth.se</acas:hasHostDomain>
					<acas:hasPhoneNumber
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">tel:+999999999</acas:hasPhoneNumber>
					<acas:hasPhoneDomain
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">1</acas:hasPhoneDomain>
				</acas:UserAddress>
			</acas:hasAddress>
		</acas:User>
</rdf:RDF>
```

<u>SergiTravelKth.owl</u>

```
<?xml version="1.0"?>
<rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
     xmlns:acas="http://localhost/OWL/userContextOntology.owl#">

		<owl:Ontology rdf:about=""/>

		<acas:User rdf:ID="sergi">
			<acas:hasContext>
				<acas:UserContext rdf:ID="sergiContext">
					<acas:hasOwner
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">sergi</acas:hasOwner>
					<acas:hasLocation
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">kth</acas:hasLocation>
					<acas:hasTask
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">travel</acas:hasTask>
					<acas:hasActivity
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">walking</acas:hasActivity>
				</acas:UserContext>
			</acas:hasContext>
```

```
                        <acas:hasAddress>
                                    <acas:UserAddress rdf:ID="sergiAddress">
                                                <acas:hasHostName
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">ccsser1.wireless.kth.se</acas:hasHostName
>
                                                <acas:hasHostDomain
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">wireless.kth.se</acas:hasHostDomain>
                                                <acas:hasPhoneNumber
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">tel:+999999999</acas:hasPhoneNumber>
                                                <acas:hasPhoneDomain
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">1</acas:hasPhoneDomain>
                                    </acas:UserAddress>
                        </acas:hasAddress>
            </acas:User>
</rdf:RDF>
```

<u>SergiLabWorking.owl</u>

```
<?xml version="1.0"?>
<rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
        xmlns:acas="http://localhost/OWL/userContextOntology.owl#">

            <owl:Ontology rdf:about=""/>

            <acas:User rdf:ID="sergi">
                        <acas:hasContext>
                                    <acas:UserContext rdf:ID="sergiContext">
                                                <acas:hasOwner
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">sergi</acas:hasOwner>
                                                <acas:hasLocation
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">kth</acas:hasLocation>
                                                <acas:hasTask
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">thesis</acas:hasTask>
                                                <acas:hasActivity
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">working</acas:hasActivity>
                                    </acas:UserContext>
                        </acas:hasContext>
                        <acas:hasAddress>
                                    <acas:UserAddress rdf:ID="sergiAddress">
                                                <acas:hasHostName
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">ccsser1.wireless.kth.se</acas:hasHostName
>
                                                <acas:hasHostDomain
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">wireless.kth.se</acas:hasHostDomain>
                                                <acas:hasPhoneNumber
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">tel:+999999999</acas:hasPhoneNumber>
                                                <acas:hasPhoneDomain
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">1</acas:hasPhoneDomain>
                                    </acas:UserAddress>
                        </acas:hasAddress>
            </acas:User>
</rdf:RDF>
```

SergiLabWc.owl

```
<?xml version="1.0"?>
<rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
        xmlns:acas="http://localhost/OWL/userContextOntology.owl#">

            <owl:Ontology rdf:about=""/>

            <acas:User rdf:ID="sergi">
                        <acas:hasContext>
                                    <acas:UserContext rdf:ID="sergiContext">
                                                <acas:hasOwner
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">sergi</acas:hasOwner>
                                                <acas:hasLocation
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">kth</acas:hasLocation>
                                                <acas:hasTask
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">thesis</acas:hasTask>
                                                <acas:hasActivity
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">wc</acas:hasActivity>
                                    </acas:UserContext>
                        </acas:hasContext>
                        <acas:hasAddress>
                                    <acas:UserAddress rdf:ID="sergiAddress">
                                                <acas:hasHostName
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">ccsser1.wireless.kth.se</acas:hasHostName
>
                                                <acas:hasHostDomain
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">wireless.kth.se</acas:hasHostDomain>
                                                <acas:hasPhoneNumber
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">tel:+999999999</acas:hasPhoneNumber>
                                                <acas:hasPhoneDomain
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">1</acas:hasPhoneDomain>
                                    </acas:UserAddress>
                        </acas:hasAddress>
            </acas:User>
</rdf:RDF>
```

SergiLabCoffee.owl

```
<?xml version="1.0"?>
<rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
        xmlns:acas="http://localhost/OWL/userContextOntology.owl#">

            <owl:Ontology rdf:about=""/>

            <acas:User rdf:ID="sergi">
```

```
			<acas:hasContext>
				<acas:UserContext rdf:ID="sergiContext">
					<acas:hasOwner
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">sergi</acas:hasOwner>
					<acas:hasLocation
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">kth</acas:hasLocation>
					<acas:hasTask
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">thesis</acas:hasTask>
					<acas:hasActivity
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">coffee</acas:hasActivity>
				</acas:UserContext>
			</acas:hasContext>
			<acas:hasAddress>
				<acas:UserAddress rdf:ID="sergiAddress">
					<acas:hasHostName
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">ccsser1.wireless.kth.se</acas:hasHostName
>
					<acas:hasHostDomain
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">wireless.kth.se</acas:hasHostDomain>
					<acas:hasPhoneNumber
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">tel:+999999999</acas:hasPhoneNumber>
					<acas:hasPhoneDomain
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">1</acas:hasPhoneDomain>
				</acas:UserAddress>
			</acas:hasAddress>
		</acas:User>
</rdf:RDF>
```

SergiLunchLunching.owl

```
<?xml version="1.0"?>
<rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#"
	xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
	xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
	xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
	xmlns:acas="http://localhost/OWL/userContextOntology.owl#">

		<owl:Ontology rdf:about=""/>

		<acas:User rdf:ID="sergi">
			<acas:hasContext>
				<acas:UserContext rdf:ID="sergiContext">
					<acas:hasOwner
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">sergi</acas:hasOwner>
					<acas:hasLocation
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">kth</acas:hasLocation>
					<acas:hasTask
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">lunch</acas:hasTask>
					<acas:hasActivity
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">lunching</acas:hasActivity>
				</acas:UserContext>
			</acas:hasContext>
			<acas:hasAddress>
```

```
                                        <acas:UserAddress rdf:ID="sergiAddress">
                                                    <acas:hasHostName
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">ccsser1.wireless.kth.se</acas:hasHostName
>
                                                    <acas:hasHostDomain
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">wireless.kth.se</acas:hasHostDomain>
                                                    <acas:hasPhoneNumber
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">tel:+999999999</acas:hasPhoneNumber>
                                                    <acas:hasPhoneDomain
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">1</acas:hasPhoneDomain>
                                        </acas:UserAddress>
                              </acas:hasAddress>
                    </acas:User>
</rdf:RDF>
```

<u>SergiClassListening.owl</u>

```
<?xml version="1.0"?>
<rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
        xmlns:acas="http://localhost/OWL/userContextOntology.owl#">

            <owl:Ontology rdf:about=""/>

            <acas:User rdf:ID="sergi">
                      <acas:hasContext>
                                <acas:UserContext rdf:ID="sergiContext">
                                          <acas:hasOwner
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">sergi</acas:hasOwner>
                                          <acas:hasLocation
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">kth</acas:hasLocation>
                                          <acas:hasTask
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">class</acas:hasTask>
                                          <acas:hasActivity
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">listening</acas:hasActivity>
                                </acas:UserContext>
                      </acas:hasContext>
                      <acas:hasAddress>
                                <acas:UserAddress rdf:ID="sergiAddress">
                                          <acas:hasHostName
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">ccsser1.wireless.kth.se</acas:hasHostName
>
                                          <acas:hasHostDomain
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">wireless.kth.se</acas:hasHostDomain>
                                          <acas:hasPhoneNumber
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">tel:+999999999</acas:hasPhoneNumber>
                                          <acas:hasPhoneDomain
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">1</acas:hasPhoneDomain>
                                </acas:UserAddress>
                      </acas:hasAddress>
            </acas:User>
</rdf:RDF>
```

SergiClassBreak.owl

```
<?xml version="1.0"?>
<rdf:RDF xmlns:owl="http://www.w3.org/2002/07/owl#"
	xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
	xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
	xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
	xmlns:acas="http://localhost/OWL/userContextOntology.owl#">

		<owl:Ontology rdf:about=""/>

		<acas:User rdf:ID="sergi">
			<acas:hasContext>
				<acas:UserContext rdf:ID="sergiContext">
					<acas:hasOwner
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">sergi</acas:hasOwner>
					<acas:hasLocation
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">kth</acas:hasLocation>
					<acas:hasTask
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">class</acas:hasTask>
					<acas:hasActivity
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">break</acas:hasActivity>
				</acas:UserContext>
			</acas:hasContext>
			<acas:hasAddress>
				<acas:UserAddress rdf:ID="sergiAddress">
					<acas:hasHostName
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">ccsser1.wireless.kth.se</acas:hasHostName
>
					<acas:hasHostDomain
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">wireless.kth.se</acas:hasHostDomain>
					<acas:hasPhoneNumber
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">tel:+999999999</acas:hasPhoneNumber>
					<acas:hasPhoneDomain
rdf:datatype="http://www.w3.org/2000/10/XMLSchema#string">1</acas:hasPhoneDomain>
				</acas:UserAddress>
			</acas:hasAddress>
		</acas:User>
</rdf:RDF>
```

Appendix C

Ser.cfg:

```
debug=3
fork=no
log_stderror=yes
listen=130.237.15.248          # put your server IP address here
port=5060
children=4

dns=no
rev_dns=no
fifo="/tmp/ser_fifo"
fifo_db_url="mysql://ser:heslo@localhost/ser"

loadmodule "/usr/local/lib/ser/modules/mysql.so"
loadmodule "/usr/local/lib/ser/modules/sl.so"
loadmodule "/usr/local/lib/ser/modules/tm.so"
loadmodule "/usr/local/lib/ser/modules/rr.so"
loadmodule "/usr/local/lib/ser/modules/maxfwd.so"
loadmodule "/usr/local/lib/ser/modules/usrloc.so"
loadmodule "/usr/local/lib/ser/modules/registrar.so"
loadmodule "/usr/local/lib/ser/modules/uri_db.so"
loadmodule "/usr/local/lib/ser/modules/auth.so"
loadmodule "/usr/local/lib/ser/modules/auth_db.so"
loadmodule "/usr/local/lib/ser/modules/cpl-c.so"

modparam("auth_db|uri_db|usrloc", "db_url", "mysql://ser:heslo@localhost/ser")
modparam("auth_db", "calculate_ha1", 1)
modparam("auth_db", "password_column", "password")
modparam("usrloc", "db_mode", 2)
modparam("rr", "enable_full_lr", 1)

modparam("cpl-c","cpl_db","mysql://ser:heslo@localhost/ser")
modparam("cpl-c","cpl_table","cpl")
modparam("cpl-c","cpl_dtd_file","/usr/local/etc/ser/context.dtd")

route {

        # -----------------------------------------------------------------
        # Sanity Check Section
        # -----------------------------------------------------------------
        if (!mf_process_maxfwd_header("10")) {
                sl_send_reply("483", "Too Many Hops");
                break;
        };

        if (msg:len > max_len) {
                sl_send_reply("513", "Message Overflow");
                break;
        };

        # -----------------------------------------------------------------
        # Record Route Section
```

```
            # ------------------------------------------------------------------
            if (method!="REGISTER") {
                        record_route();
            };

            # ------------------------------------------------------------------
            # Loose Route Section
            # ------------------------------------------------------------------
            if (loose_route()) {
                        route(1);
                        break;
            };

            # ------------------------------------------------------------------
            # Call Type Processing Section
            # ------------------------------------------------------------------
            if (uri!=myself) {
                        route(1);
                        break;
            };

            if (method=="ACK") {
                        route(1);
                        break;
            } else if (method=="INVITE") {
                        if(!cpl_run_script("incoming","is_stateless"))
                        {
                          # script execution failed
                          t_reply("500","CPL script execution failed");
                        };
                        route(3);
                        break;
            } else      if (method=="REGISTER") {
              #handle REGISTER messages with CPL script
              cpl_process_register();
         route(2);
              break;
            };

            lookup("aliases");
            if (uri!=myself) {
                        route(1);
                        break;
            };

            if (!lookup("location")) {
                        sl_send_reply("404", "User Not Found");
                        break;
            };

            route(1);
}

route[1] {
            # ------------------------------------------------------------------
            # Default Message Handler
```

```
        # ----------------------------------------------------------------
        if (!t_relay()) {
                sl_reply_error();
        };
}

route[2] {
        # ----------------------------------------------------------------
        # REGISTER Message Handler
        # ---------------------------------------------------------------
        sl_send_reply("100", "Trying");

        if (!www_authorize("","subscriber")) {
                www_challenge("","0");
                break;
        };

        if (!check_to()) {
                sl_send_reply("401", "Unauthorized");
                break;
        };

        consume_credentials();

        if (!save("location")) {
                sl_reply_error();
        };
}

route[3] {
        # ----------------------------------------------------------------
        # INVITE Message Handler
        # ----------------------------------------------------------------
        if (!proxy_authorize("","subscriber")) {
                proxy_challenge("","0");
                break;
        } else if (!check_from()) {
                sl_send_reply("403", "Use From=ID");
                break;
        };

        consume_credentials();

        lookup("aliases");
        if (uri!=myself) {
                route(1);
                break;
        };

        if (!lookup("location")) {
                sl_send_reply("404", "User Not Found");
                break;
        };

        route(1);
}
```

COS/CCS 2007-02