PROJECTE DE FI DE CARRERA

TÍTOL: Development of an advanced web application for managing videoconference

AUTOR: Xavier Calvo Brugal

DIRECTOR: Antoni Oller Arcas

DATA: 9 de febrer de 2007

Título: Development of an advanced web application for managing videoconference

Autor: Xavier Calvo Brugal

Director: Antoni Oller Arcas

Fecha: 9 de febrero de 2007

Resumen

Existen numerosas aplicaciones que ofrecen al usuario la posibilidad de realizar videoconferencias en alta definición sobre Internet. Estas aplicaciones centran sus esfuerzos en la transmisión de contenido de alta calidad sobre Internet, pero dejan sin resolver la gestión del establecimiento, finalización de llamadas, aceptación, rechazo de invitaciones, suscripción y recepción de notificaciones del servicio de presencia (aparición de usuarios, cambios de estado, etc).

El objetivo del proyecto es realizar un cliente para videoconferencias de alta definición basado en Web. Se ha utilizado SIP como protocolo de señalización (establecimiento, finalización) de videoconferencias, para la gestión la lista de los contactos, la presencia, y la negociación de las capacidades multimedia

La aplicación desarrollada permite que, un usuario a partir de una web, sea capaz de ver todos los usuarios conectados al servicio, conocer sus características y poder establecer videoconferencias de alta calidad, utilizando el entorno de videoconferencia negociado (por ejemplo dvts, ultragrid).

La aplicación actúa como un gateway HTTP-SIP, traduciendo las peticiones SIP a peticiones HTTP y viceversa. Las peticiones HTTP son enviadas hacia el cliente para que la trate y realice los cambios necesarios en la interfaz web y en su modelo de datos. La interfaz de web del usuario se ha desarrollado con Google Web Toolkit, un toolkit de Google para el desarrollo de aplicaciones AJAX en lenguaje de programación Java.

El servidor se comunica con tres módulos. Con el mundo SIP para la señalización de videoconferencia, con el agente de presencia para gestionar la lista y con el cliente AJAX para comunicarse con los usuarios de la aplicación.

El proyecto explica las fases de definición de requerimientos, diseño y arquitectura de la aplicación. Expone el estado del arte de las tecnologías y comenta algunos detalles de la implementación, finalizando con una planificación temporal y unos resultados finales del proyecto.

Title: Development of an advanced web application for managing videoconference

Author: Xavier Calvo Brugal

Director: Antoni Oller Arcas

Date: February, 9th 2007

Overview

Several applications offer high-definition videoconference over Internet. These applications are designed for a better performance transmitting high bit rate multimedia data, leaving in a secondary plane other important aspects as the manage of establishment, finalization of the sessions, acceptation or rejection of the invitation, subscription and notification of presence service (new users connected, change of status, …)

The aim of the project is to realize a client for high-definition videoconference based on Web. We use SIP as videoconferences signalling protocol (establish, finalize), for managing the contact list, presence and multimedia capabilities negotiation.

Developed application allows that, a user from a web can see all connected users to the service, can consult their characteristics and can be able to establish videoconferences, using videoconference environment negotiated (as example DVTS or Ultragrid).

The application acts as a HTTP-SIP gateway, translating SIP requests to HTTP requests and vice versa. HTTP requests are sent to client in order that client handles it changing view or modifying data model. The user interface has been developed with Google Web Toolkit, a toolkit of Google for developing AJAX applications in Java language.

Server-code must communicate with three modules. Communication with SIP world for videoconference signalling, communication with the presence agent to manage the contact list information and communication with AJAX client for notifications to application users.

This report explains the phases of definition of requirements, design and architecture. In addition, it is explained state-of-art of the technologies and some implementation details. Finally expose planning and final results of the project.

INDEX

INDEX OF FIGURES

INDEX OF TABLES

CHAPTER 1. INTRODUCTION

In the last years, the videoconferencing systems have been an important area of telecommunications research, resulting in a large number of products that can be used to transmit and receive video in real time over IP networks. These systems vary mainly in terms of quality and in the amount of bandwidth used to transmit video over networks. In this sense, the telecom vendors and manufacturers have focused on developing low to medium bandwidth conferencing systems typically based on the H.320 and H.323 standards. In addition to these commercial products and systems, the academic research and university community have also looked for higher quality and higher bandwidth video systems that operate over high speed research and educational networks. Two important examples of this type of video system of high quality are the following ones: Digital Video Transport System (DVTS) [1] and Ultragrid [2].

However, most of the existing systems, as DVTS and UltraGrid, are focused on how to transport high bit rate multimedia data, leaving in a secondary plane the control of them.

Powerful videoconference systems may turn out unconformable when they do not have signalling mechanisms, for example to establish a communication with your partner. Without signalling, your partner does not notice that are receiving an incoming call. You must phone him notifying your intention to establish a communication, requesting to prepare his videoconference system, and later that he should have to pick up the phone, camera, microphone… Obviously, this situation may be avoided with signalling mechanism.

SIP (Session Initiation Protocol) [6] is one of most popular protocols in Internet community for creating, modifying, and terminating telephone calls, multimedia distribution, and multimedia conferences. It is used in Commercial IP Telephony for example.

This thesis is inside of “Machine” Project. Machine is a research project of i2Cat Foundation [7] that corresponds to the second phase of “Projecte Integrat”, which also carried out by the same organization last year. Work points PT2 and PT3 describe the task to develop a SIP client to control videoconferences of low, medium and high quality and streaming. Another objective is to get an easy graphic user interface, which compose the maximum services.

In this sense, a first release of the SIP client has been developed and tested successfully with DVTS and Ultragrid. This first SIP client received orders by command line. It could establish a DVTS or Ultragrid session between two user agents, using a SIP Server.

Now a beta graphic SIP client, based in Java Swing, shows a fixed list of users (loaded from a XML). Its features are establish/finish sessions, incoming call notification and accept/refuse this call.

Along time of developing of this project, graphic SIP client has been improved with some feature, as example, presence service or DVTS software integration.

1.1. Objectives

The main objective of this project is to add a signalling layer to high definition videoconference systems and make a simple interface to manage signalling. For this propose, we must develop a graphic user interface for the SIP client. This interface must collect different services and as possible without new software to install in client machine. As example of integrate environment may be Gmail application of Google enterprise. Gmail offers by itself, without external programs only a web browser, e-mail, search engine and instant messenger with presence service, user information and some more features.

Web applications are the most comfortable, easiest and ubiquitous, according opinion polls. Therefore one of the main characteristic of this design will be that can be accessed via web. Consequently, we need to research available technologies to approach that.

We are innovating adding signalization to high-definition conference system, hiding complexity to user, making all process transparent. One aim of this project is to integrate a SIP Client with our interface to a multimedia platform. Moreover, this project can become an example of HTTP-SIP convergence, an unexperimented field.

This project has been structured as following way. First chapter introduces the project inside a research framework. Second chapter defines features of the application to develop. Next two sections study system architecture and how our project has been adapted and modelled. Fifth chapter describes all used technologies and several details about our implementation. Sixth section estimates overall cost of project and details how has been planed. Finally we close balance about achieved objectives, propose future works on same direction and relate our experience.

CHAPTER 2. SPECIFICATIONS

Our aim is design an application capable to manage a SIP Client, accessible via web, and with multimedia support. On this chapter have been described principal features required for expected use.

2.1. Features

Web access

Application should be able to be accessed via web with a common web browser. Web access ensures that may access to application from anywhere.

Authentication

For joining to application, user must have a registered account. At the beginning of every session, application asks for his account

Edit your profile

User can edit his public information. It is possible to modify state, multimedia capabilities and some other information. First release only considers two multimedia profiles, DVTS for medium quality and Ultragrid for high quality. However, system has to be able to support new multimedia formats in future. These systems must be added as pluggable services and their characteristics will have to be configured.

Contact list

Application must keep updated connected user list. Thought this list we access to user panel, which shows contact description and is possible to interact with them. When others contacts connect or change its states, changes will be reflected on list. Behind contact list works a presence service, which shares contacts information.

Videoconference management

Application must be able to initiate videoconferencing with one of his contacts, receive invitation to participate in one, accept or reject this invitation and finalize active videoconference. Moreover may be able to select the videoconference's mode. That all for every multimedia system supported by application.

Minimal installation required

Ideally, user does not need install external software to enjoy this product. Obviously, minimal requirements may force us to demand some software installed like a web browser.

Compatibility

System designed must to be compatibility with existing applications (Swing client), and must integrated with rest of services.

Platform Independence

Business logic may be portable to other platform without modify code application. Only condition is that new platform must dispose a Java Virtual Machine. Client code also must be executed over different web browser and different platform.

2.2. Formal specifications

Fig. 2.1 shows the main functionalities that offers our web interface.

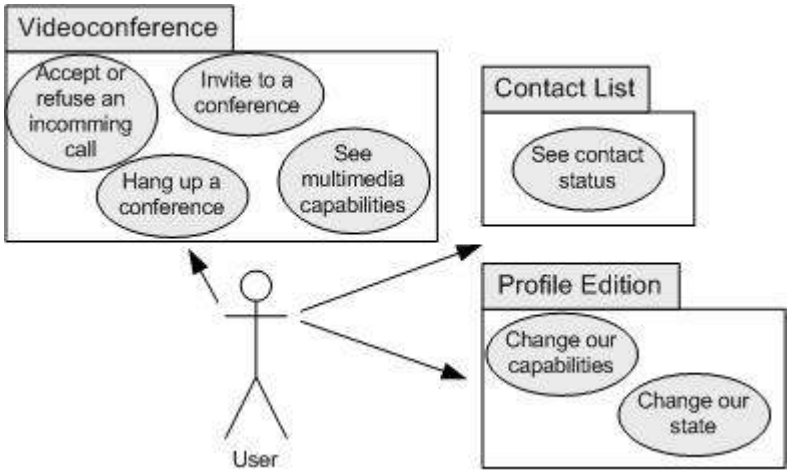

Fig. 2.1 Use diagram

Following tables describe every use case with a short descriptions, it normal flow and what conditions may accomplish to be executed.

2.2.1 Profile Edition

Name	Change capabilities
Description	Modify our multimedia capabilities notifying rest of contact the news abilities
Actors	User
Preconditions	User has been logged successfully and subscribed to presence agent.
Normal flow	1. User opens configuration panel. 2. User marks his available capabilities. 3. Click on Confirm button.
Alternative flow	Automatic detection (not proposed)
Input	News capabilities
Output	Notification to presence service.

Name	Change our status
Description	User modify his current status and notifies rest of contact.
Actors	User
Preconditions	User has been logged successfully and subscribed to presence agent.
Normal flow	1. User opens configuration panel. 2. Change to desired state. 3. Click on Confirm button.
Input	New status
Output	Notification to presence service.

Name	See multimedia capabilities
Description	Thought the contact panel may be possible to see all multimedia capabilities of contact selected
Actors	User
Preconditions	User has been logged successfully and subscribed to presence agent.
Normal flow	1. User opens contact panel.
Output	Visual information about capabilities.

2.2.2 Videoconference

Name	Invite to a conference
Description	Invite a contact to establish a videoconference
Actors	User
Preconditions	User has been logged successfully. Callee must be connected User and callee may have compatibles multimedia modes
Normal flow	1. User opens contact panel. 2. Select videoconference mode. 3. Click on Invite button.
Post conditions	If other extreme accept invitation, videoconference start automatically.
Input	Videoconference mode
Output	Invite to selected user.

Name	Hang up a conference
Description	Hang up an establish videoconference
Actors	User
Preconditions	Must exist an active videoconference
Normal flow	1. User opens videoconference panel. 2. Click on Hang up button.
Alternative flow	Videoconference may be closed by other extreme
Output	It closes all the related software (dvts xdvshow, ultragrid)

Name	Accept or refuse an incoming call
Description	When you receive a videoconference invitation, you can accept it or discard it. If invitation is accepted videoconference start.
Actors	User
Preconditions	User had to have received an invitation.
Normal flow	1. Appear a popup menu with incoming call description 2. Click on Accept or Refuse button.
Post conditions	When a videoconference is active, you do not be able to initiate another conference.
Input	Invite message from caller.
Output	Response to caller and if is necessary runs all necessary software for videoconference

2.2.3 Contact List

Name	See contact status
Description	Thought the list may be possible to see status of other users. (online, offline, …)
Actors	User
Preconditions	User has been logged successfully and subscribed to presence agent.
Normal flow	1. Contact list must have a visual icon which represents current status of this contact
Alternative flow	This information is represented too in contact panel,.
Output	Visual information about all contact list.

CHAPTER 3. ARCHITECTURE

The simplest scenario of our system (Fig. 3.1) is formed by two clients on PCs, a SIP Server (SER) [8] and a Presence Agent. Black arrows (I, II, III,…) describe dialog between a client and SIP Server. Blue arrows (1,2,3,…) represent dialog between a client and presence agent.

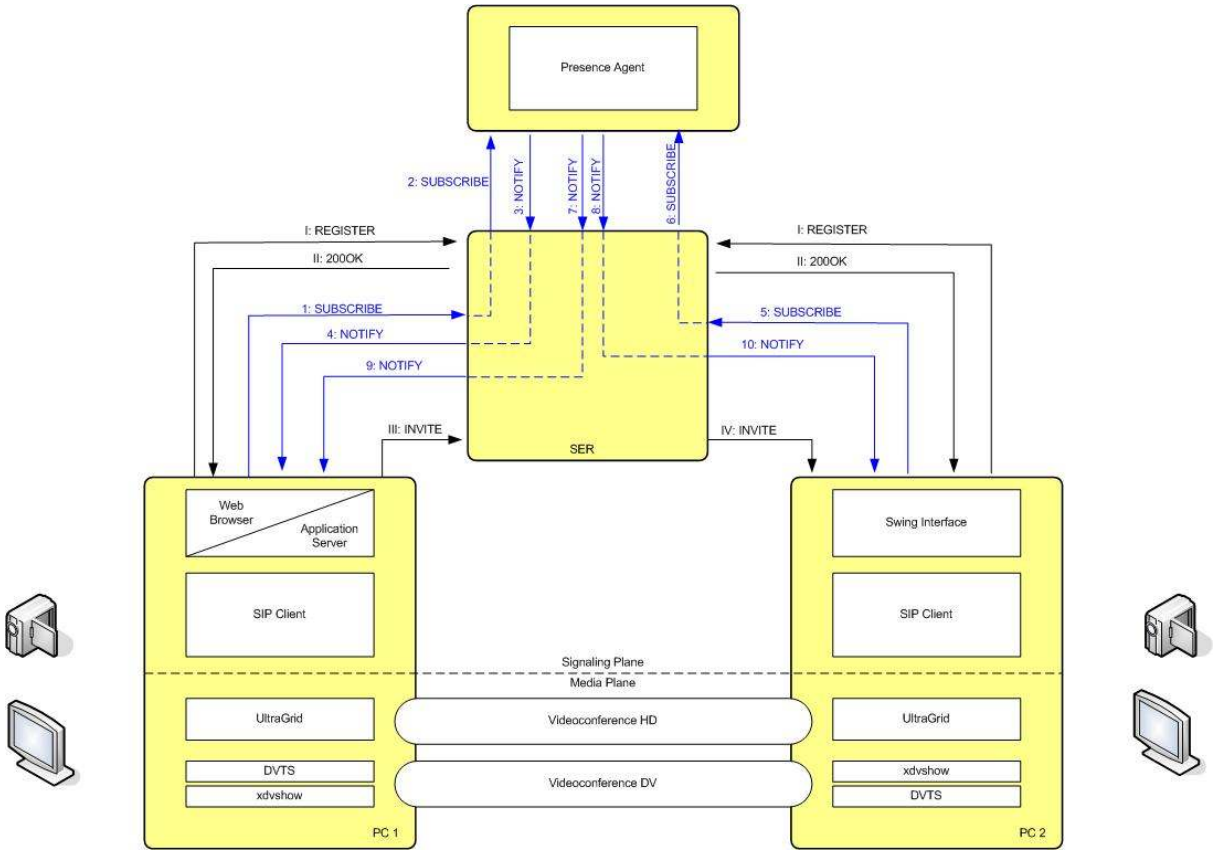

Fig. 3.1 Global system architecture

In system architecture, we can distinguish two planes, signalling plane and media plane. Media plane is composed of different available videoconference mechanisms. In our project, this plane is less important than signalling plane because our contribution is developing a signalling mechanism for videoconference environment.

We use SIP for signalling proposes, and SIP Client is in charge to send and receive SIP message, for these reason SIP Client is the main piece of signalling plane. For controlling it, we have two graphic user interfaces, one based on Java Swing and other based on AJAX technology [3].

A characteristic of this scenario is that follows philosophy of SIP-based Continuous Media Integration (SIP-CMI) [4]. SIP-CMI platform is an open, flexible, scalable testbed to support a wide and extensible set of next-generation continuous media services. This platform follows the principle that any continuous media service can be accessed by using the SIP protocol,

regardless of the nature of the service; for example videoconference or streaming.

On the other hand, there is a presence service. Its function consists on informing to all clients about status of contact list. It is a simplified SIP client with two tasks. Receiving subscribe message when a new user connects to the system or changes his user parameters. Every subscribe message, therefore a modification on contact list, presence agent sends a notify message with new contact list.

In this project, we concentrate our efforts towards developing a client with interface via Web and integrate to application all signalling plane, formed by SIP Client, presence service and videoconference modules.

3.1. Application architecture

All web services follow client-server architecture (Fig. 3.2.) Client side is the user machine. It only has a web browser, which shows HTML pages and interprets Javascript code. This Javascript code is the AJAX Client, which communicates with application server.

Server side may be an external machine, independent of the client. Here is where we place an application server, as example Tomcat, which runs our application code. In this case, application code can be simplified as a servlet, which translates HTTP to SIP and vice versa. SIP messages are delivered to SIP Client.

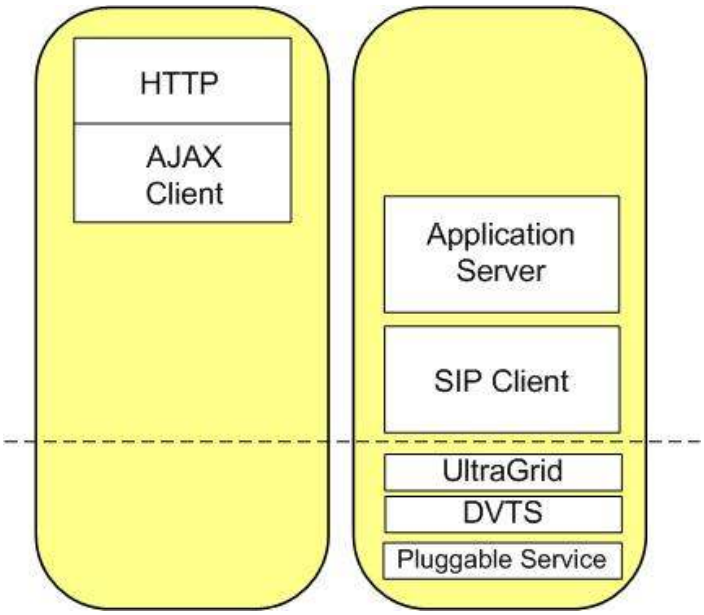

Fig. 3.2 Client architecture

Multimedia plane corresponds to all software and connections for transmitting, receiving and showing videoconferences streams.

SIP client of the signalling plane must open videoconference program of the multimedia plane. To open external programs from a browser is critical issue due to security restrictions. As a first approach, client and server will be in the same machine, and application server will open videoconference software. For this reason, in Fig. 3.2 multimedia technologies are in server-side. Possible solutions for this issue commented in section 5.2.5

Current software philosophy follows the paradigm of a unique and complex font and the more doom clients as are possible. Therefore, interest that client architecture become simplest as be possible, and complexity place on source. But in HD software requires powerful machine, opposite to ideal scenario. For this reason, we try to separate multimedia plane from signalling plane, proposing to do “black box” optimized for videoconference purposes.

3.2. Overview of scenario

On previous sections have been motioned signalling plane, multimedia plane and several actors as users, presence agent or SER [8]. In Fig. 3.3 we observe all actor and his paper on every plane. Presence Agent and SIP Proxy (SER) only work in signalling plane, they do not transmit any multimedia stream, therefore they do not any module in multimedia plane. Otherwise, clients have to interact in all planes.
User plane is the physic representation of every piece.

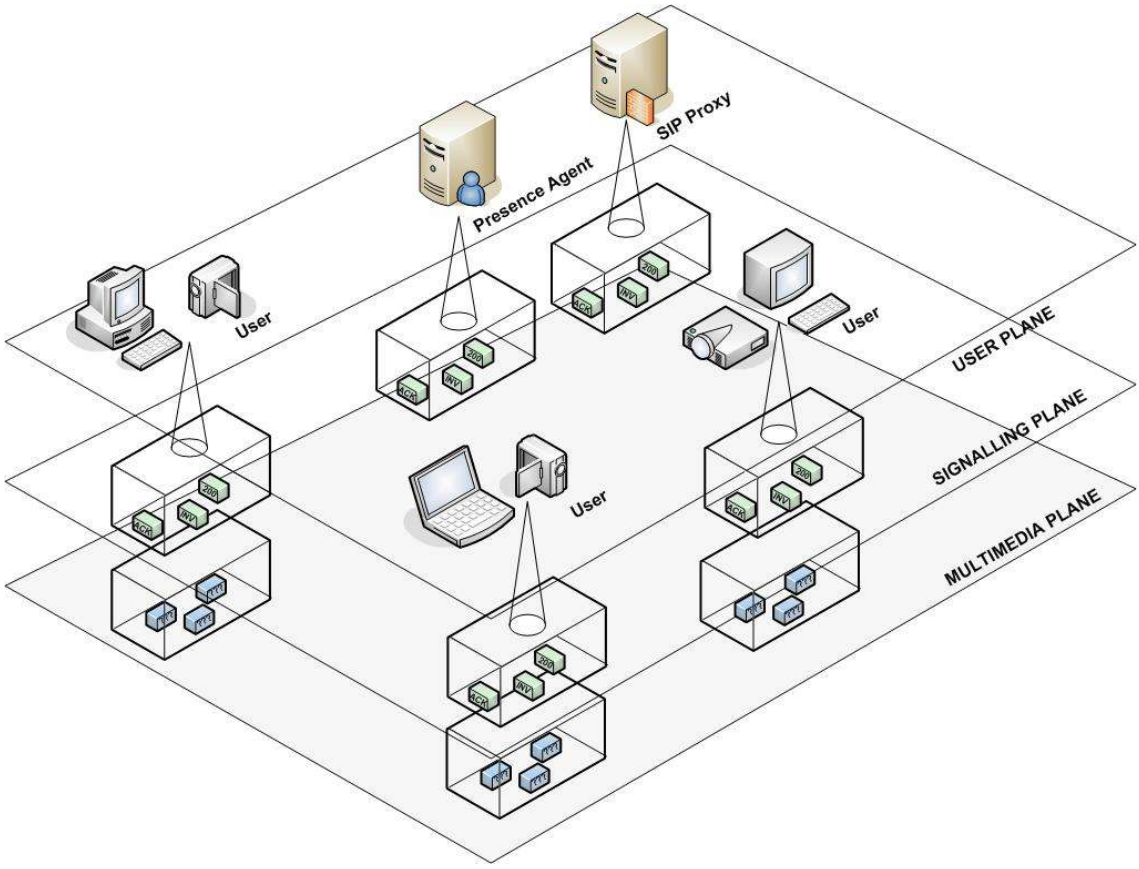

Fig. 3.3 Layers and actors

CHAPTER 4. SYSTEM DESIGN

There are several interesting pieces to comment their designs. In this chapter we explain deeply communication model between sever and client, especially from server to client.

4.1. Data Model

One task of this application is keep updated the contact list. This list is formed by groups. In every group, we can find several users and each user may support different multimedia types as DVTS or Ultragrid.

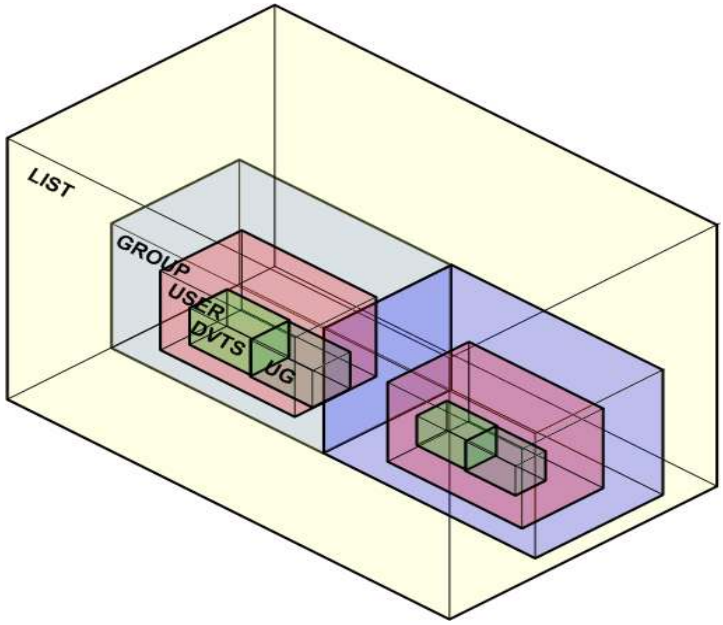

Fig. 4.1 Data model

Contact list is a collection of users. This data structure is the most important information shared between client and server. User is the minimal structure with relevant importance in our application. A user is defined by following attributes:

User-ID
: Unique identifier for each user.

Nick
: Name of user. Usually matches with userID

State
: Information about state. A user can be online, offline, busy…

Multimedia capabilities
: User publishes which are his multimedia capabilities. This description depend of every multimedia mechanism and which parameters can be configured, by example number of frames per second, or codec supported.

4.2. Communication Model

Client and server machines must keep an active connection for exchange messages. AJAX interface make all transaction between client and server. It prepares information for be sent across network and in other side, it receive and deliver information to upper module.

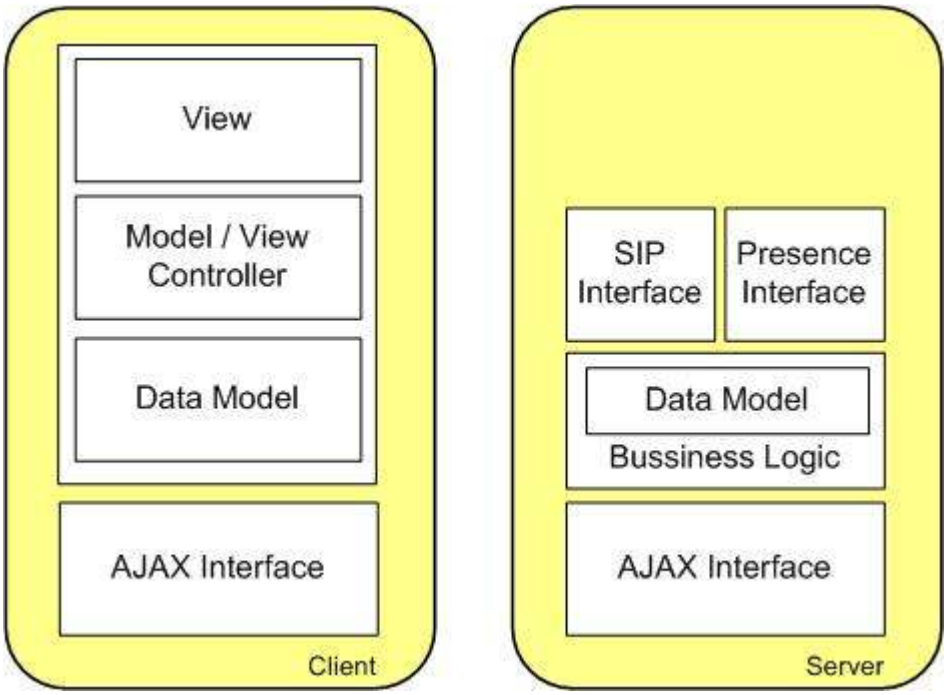

Fig. 4.2 AJAX Design

Client machine follows Model-View-Controller pattern [9]. In our application is important to separate data and view, and it is done introducing an intermediate component, the Controller.

Data model is the representation of the information on which the application operates.
View offers to user an abstraction of data container, representing it into visual forms, which allow interaction from user. View accepts inputs, allowing the user to interact with the application, and output, allowing the system to notify the effects of the users' manipulation or external events.
Controller processes and responds to events, typically user actions or AJAX events, and may invoke changes on the model, changing sometimes current view, but every event does not imply a change on model or view.

On server side, is a similar case that client, but without view. There is a controller module, which changes information model. In this case, controller interacts with several pieces. AJAX interface has the same function than on client side. Now appears two news interfaces, SIP interface and presence interface. Our application acts as HTTP-SIP gateway, SIP interface is our gateway towards SIP world. In other hand, we have got a presence service,

also in SIP but managed in different manner, for this reason we need presence interface.

4.2.1 AJAX Interface

AJAX interface have two working types, communication from user to server and form server to user.

4.2.1.1. User to server communication

This type of communication happens usually when user interacts with web interface. User can do two actions that are following described.

User profile modification

When user changes his profile, user must notify new user state, sending changes to server. In addition, web client can request information about a contact, or other type of information about contact list. We must to define the interface for attending this kind of request and reply it.

Videoconference manage

All message related to videoconference managing, as accept/refuse an incoming call or close a conference.

4.2.1.2. Server to user communication

These kinds of communications are more difficult to predict. They are generated by SIP events or presence events. An extra difficulty is that server cannot send asynchronous message to client; previously server needs a request message from client. Next list shows the reasons because server may send message to client.

SIP events

All incoming message to client about videoconference managing.

Presence events

Presence agent sends events every time list changes. This message has to be delivered to client.

Client response

Reply to request message. This case is easier, because we have now a request message.

4.2.2 SIP Interface

SIP Client is a legacy piece of project therefore, we do not explain on detail every feature of it. Moreover, SIP Interface is a fixed interface, which we do not modify anything, even though some new features have been added, for example, negotiation sessions into invite's messages. This new features has been designed in collaboration with other members.

Basic dialogs on SIP are establish dialog and disconnect dialog

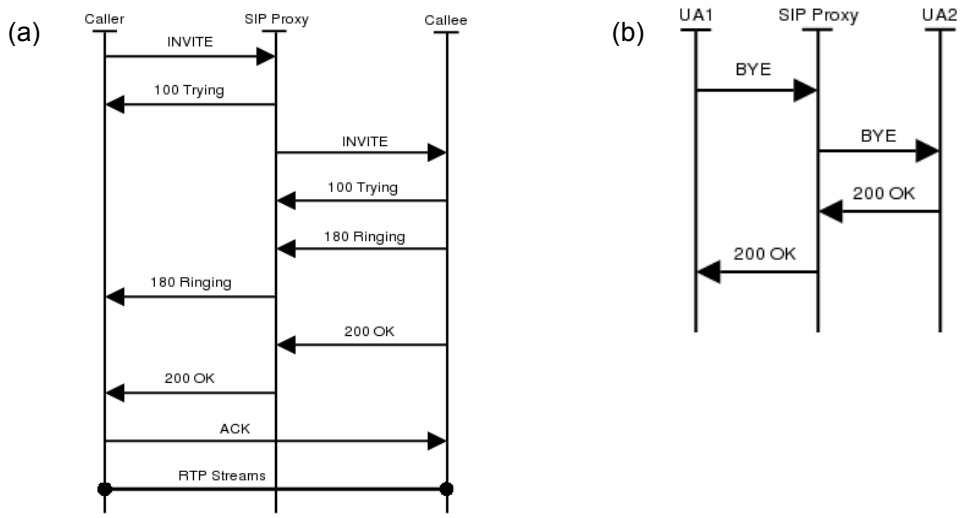

Fig. 4.1 Establish dialog (a) and disconnect dialog (b)

SIP interface allows send and receive basic message to initiate/close sessions, accept/refuse invitation.

We have to implement SIP interface. Our objective is to achieve a behaviour similar than Fig. 4.2, where SIP interface acts as a SIP Gateway.

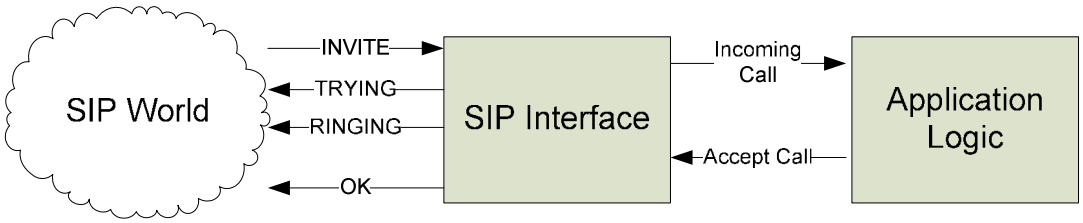

Fig. 4.2 SIP Interface behaviour

4.2.3 Presence Interface

Presence interface is our connector with presence service. This service keeps contacts status and publishes complete list to all clients. Presence interface must be able to subscribe to presence service and receive notify message.

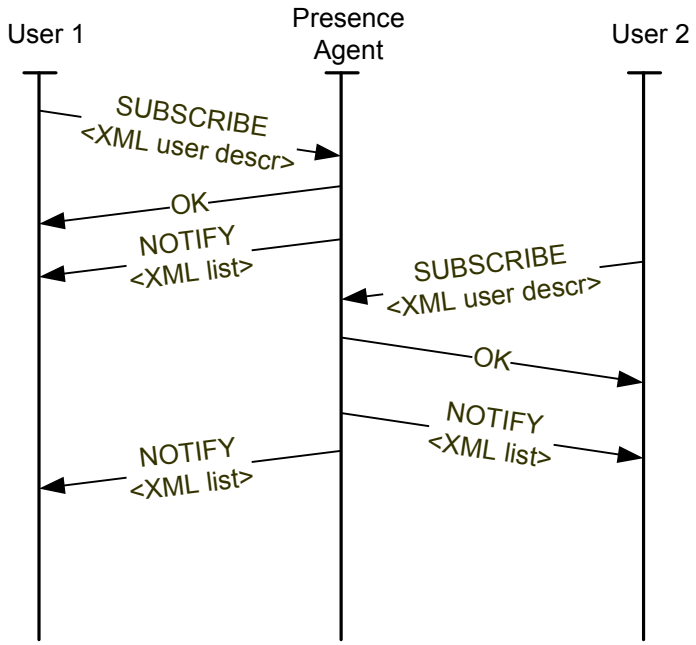

Fig. 4.3 Presence Dialog

A user subscribes to agent presence every time it connects to the application or modifies his characteristics.
On subscribe message is attached an XML like Table 4.1. XML message contains all information about user. User id, multimedia capabilities,…

Table 4.1 Subscribe XML

```
<user>
        <id>user1</id>
        <dvts tx=“true" rx=“true“    ip="192.168.48.164"/>
        <ug tx=“false" rx="false“    ip="192.168.48.164"/>
</user>
```

Our application will receive notify messages every time contact list change, every subscribe sent by any contact. Notify message attaches an XML like Table 4.2 that represent complete list.

Table 4.2 Notify XML

```
<contacts>
        <group>
                <name>connected group</name>
                <users>
                        <user>
                                <id>user1</id>
                                <dvts tx=“true" rx=“true“
                                        ip="192.168.48.164"/>
                                <ug tx=“false" rx="false“
                                        ip="192.168.48.164"/>
                        </user>
```

```
                    <user>
                        …
                    </user>
                </users>
            </group>
</contacts>
```

We have to implement Presence interface to achieve a behaviour similar than Fig. 4.3. In this case, we have to manage XML code and parse to Java class.

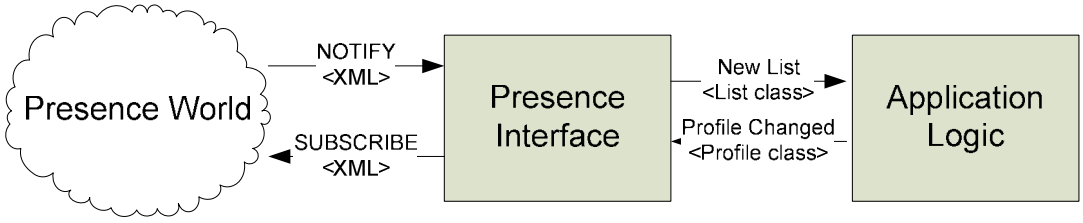

Fig. 4.3 Presence Interface Behaviour

Design of Presence service, presence communication system, and XML format has been done in collaboration with other members.

4.3. Persistent connection

All services based on HTTP follows Client-Server architecture. This architecture defines client as active part who initiate connection sending request and waiting for server reply. (Table 4.3)

Table 4.3 Client-Server role

Server	Client
Passive (slave)	Active (master)
Waits for requests	Sends requests
Upon receipt of requests, processes them and then serves replies	Waits for and receives server replies

AJAX improves client functions sending asynchronous request and getting better network performance, but server still is a passive actor.

We need an active server, which should be able to notify to client with event message. Ideally, we want have a persistent channel where transmit all communications. This demand differs from client-user architecture, where server cannot initiate any connection.

For achieving that client would receive notifications from server there are two possible scenarios

1. Polling. Client polls server every *X* seconds about changes.
2. Push emulation. Server cannot send anything without previous request from client. We can hold up client's request a short time until we have anything to reply.

First possibility is easier to implement than second but its efficiency is lower. Therefore, we choose second one. Push concept is based on sending when it has anything to serve, asynchronous and without waiting time. Fig. 4.4 show how emulate "push behaviour".

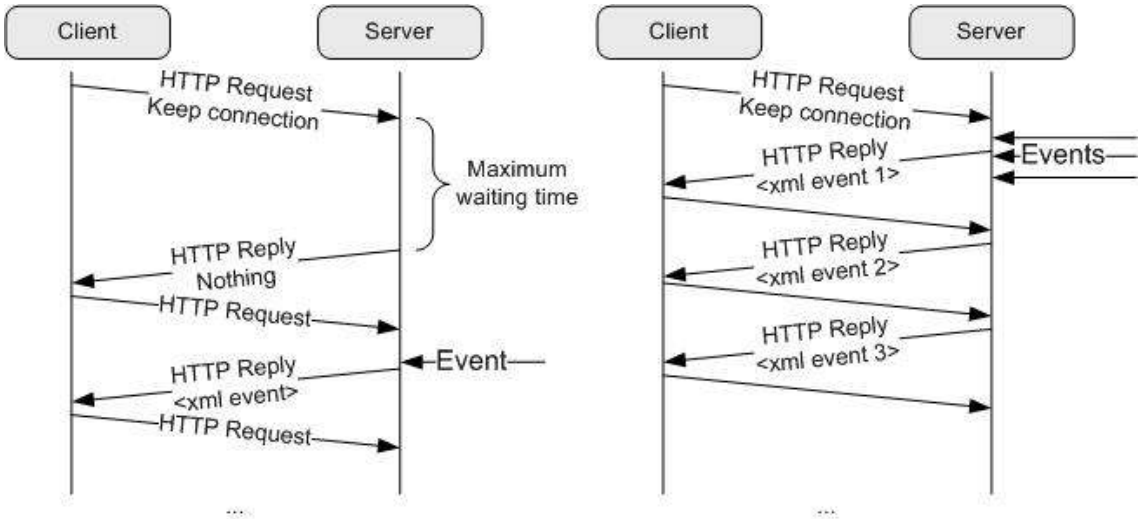

Fig. 4.4 Persistent connection dialogs

This solution has two weak points:

1. Persistent connection can break if client message is lost.
2. Compromise between connection timeout and amount time to hold up request.

Fig. 4.5 represents logic algorithm of persistent connection. In this diagram, we emphasize two points. First, there is a container for events, this box is going to fill by different reasons, and this mechanism empties sending every event in order. Second detail is limit time control. If this timeout expires connexion is returned without response.

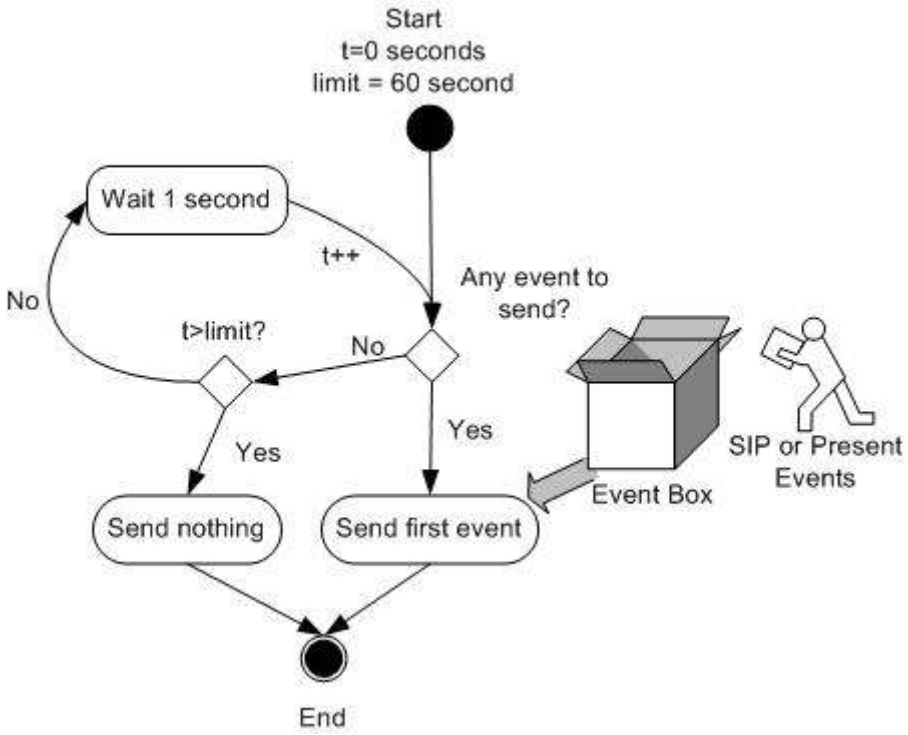

Fig. 4.5 Persistent connection activity

4.4. Application flow

Fig. 4.6 (next page) shows main interactions between client-side and server-side. Moreover, in diagram we can observe that server-side acts mainly as SIP gateway, translating SIP messages to applications messages and vice versa but sometimes it has to take some decisions.

However, client-side is not a dummy-client, on the contrary it have great amount of logical code.

Diagram shows:

- User joins to application and authenticates
- His contact list is updated
- Call to a contact
- Receive an incoming call
- Start/Finish a videoconference

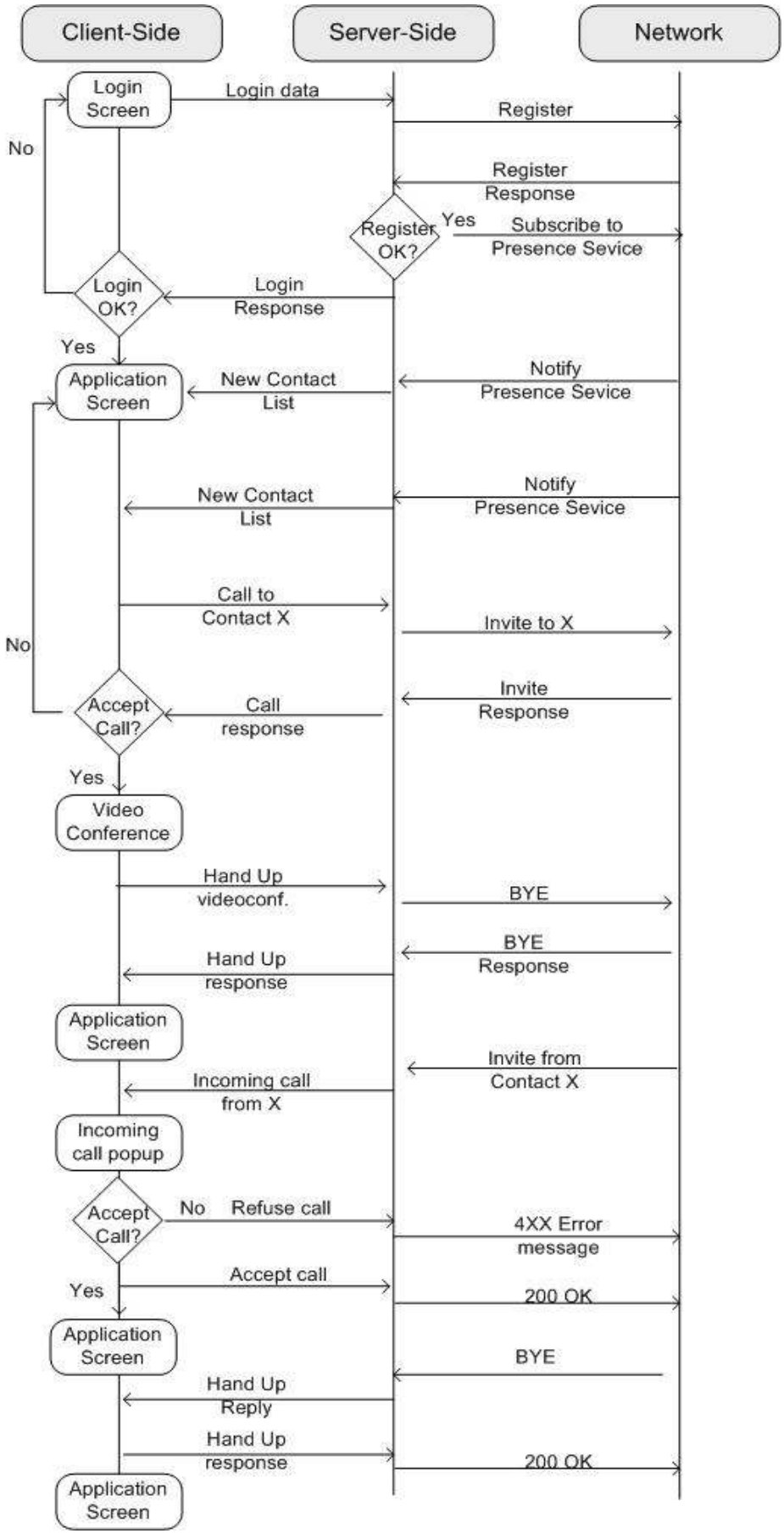

Fig. 4.6 Application flow diagram

CHAPTER 5. IMPLEMENTATION AND TESTING

This chapter describes our develop scenario, technologies and tool used, some details about implementation phase, and finally testing and integration sequence.

5.1. Scenario

Our scenario is formed by two clients PC, one proxy/register server and another machine, which runs Presence Agent. (Fig. 5.1)

Two clients PC characteristics are

- AMD Athlon™ 64 3800+
- 1 GB RAM memory DDRII 400Mhz
- Video card NVIDIA® GeForce™ 7300LE Turbocache
- Motherboard Chipset Nvidia GeForce 6150LE
- Hard disk IDE 250 GB
- Ubuntu Linux 6.06 LTS

SIP Sever and Presence Agent machine characteristics are:

- Intel Celeron 2400
- 500 MB RAM memory DDRII
- Hard disk IDE 40 GB
- Windows XP Professional SP2 or Ubuntu Linux 6.06 LTS

All of them are connected to an 8-ports Gigabit switch of LongShine manufacturer [10].

Client PC may be equipped with DV camera. We use two cameras, one for each client, connected by Firewire port.

Cameras used are:

- Canon MiniDV Optura10. NTSC Camera.
- Panasonic PAL DV Camera.

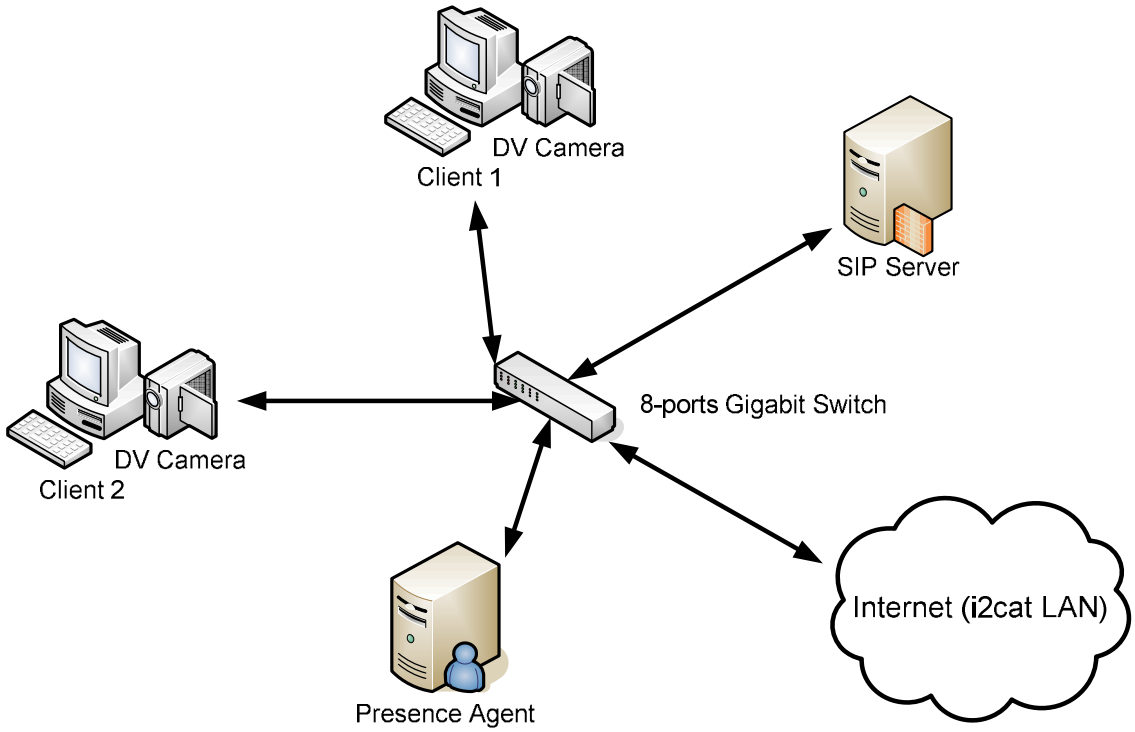

Fig. 5.1 Scenario

5.2. Technologies and tools

This section enumerates and describes the different technologies and tools used by this project.

JDK 5.0: Java Virtual Machine and Development Kit used. It is used the different libraries that it provides in order to develop the Java-based application [11] .

Eclipse SDK 3.1: Open development and compilation platform for Java comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. A large and vibrant ecosystem of major technology vendors, innovative start-ups, universities, research institutions and individuals extend, complement and support the Eclipse platform. Its basic features can be extended through plug-ins. The whole application has been developed using this development platform [12]

Log4j: Java-based logging utility. The main criterion for introducing a logging system to the application is that it helps with debugging tasks. Furthermore, it is especially useful for distributed application such as the one proposed in this project.

With log4j it is possible to enable logging at runtime without modifying the application binary. The log4j package is designed so that these statements can remain in shipped code without incurring a heavy performance cost. Logging behaviour can be controlled by editing a configuration file, without touching the application binary. [13]

SAX (Simple API for XML): Serial access parser API for XML. SAX provides a mechanism for reading data from an XML document. It is a popular alternative to the Document Object Model (DOM). [14]

Apache Ant 1.6: Java-based build tool. It is like a Makefile when talking about a C environment. It allows a lot of tasks such as compile, execute, copy files, create directories, create JAR/WAR files, run unitary tests, and so on. It uses XML syntax and it is platform independent. [15]

CVS (Concurrent Version System): Implements a version control system: it keeps track of all work and all changes in a set of files, typically the implementation of a software project, and allows several (potentially widely separated) developers to collaborate. CVS has become popular in the free software and open-source worlds.

5.2.1 AJAX

5.2.1.1. Ajax definition

Ajax, shorthand for Asynchronous JavaScript and XML, is a web development technique for creating interactive web applications. The intent is to make web pages feel more responsive by exchanging small amounts of data with the server behind the scenes, so that the entire web page does not have to be reloaded each time the user requests a change. This is meant to increase the web page's interactivity, speed, and usability.

The Ajax technique uses a combination of:

- XHTML (or HTML) and CSS, for marking up and styling information.
- JavaScript to dynamically display and interact with the information presented.
- XML is used as the format for transferring data between the server and client, although any format will work.

5.2.1.2. Ajax toolkit research

Some AJAX frameworks are appearing in market. At beginning of this project, we research and evaluate Dojo [16], Mochikit [17], Google Web Toolkit (GWT) [18].

A few years ago, for developing an AJAX application was needed to program in JavaScript language. This language does not have comfortable developing environments', neither was possible to debug code. In short, programming in JavaScript language was a tedious task.

Dojo and Mochikit are JavaScripts libraries or toolkits that provide several implemented JavaScript functions.

GWT, shorthand for Google Web Toolkit, is an open-source toolkit by Google to develop AJAX applications in the Java programming language. GWT supports rapid client/server development and debugging in any Java IDE.

5.2.1.3. Google Web Toolkit

Google Web Toolkit (GWT) is an open source Java development framework that lets us escape the matrix of technologies that make writing AJAX applications so difficult and error prone. With GWT, we can develop and debug AJAX applications in the Java language using the Java development tools of our choice. When we deploy our application to production, the GWT compiler translates Java application to browser-compliant JavaScript and HTML.

Here is the GWT development cycle:

1. We write and debug our application in the Java language using our favourite Java IDE, using as GWT libraries.
2. Use GWT's Java-to-JavaScript compiler to distill application into a set of JavaScript and HTML files that we can serve with any web server.
3. Confirm that application works in each browser that we want to support, which usually takes no additional work.

We have decided to use GWT by following reasons:

- Java technologies offer a productive development platform more powerful than JavaScript.
- GWT asserts us compatibility with principal web browsers
- Many workgroups are developing applications over GWT and a great community discuss daily about GWT.
- GWT has some widgets to design user interface.
- Two killer applications as Google Maps [19] and Gmail [20] have been developed with GWT.

5.2.1.4. Google Web Toolkit architecture

GWT has four major components: a Java-to-JavaScript compiler, a "hosted" web browser, and two Java class libraries:

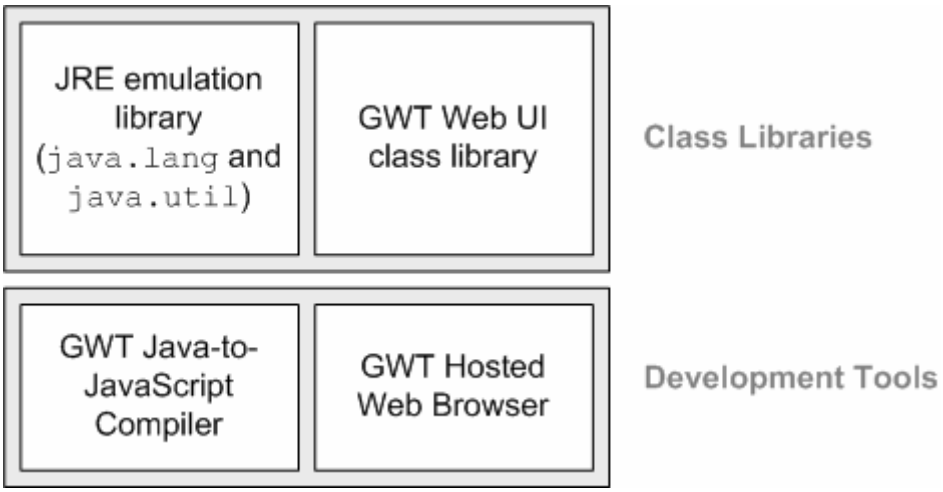

Fig. 5.2 GWT architecture

The components, from bottom to top, are:

GWT Java-to-JavaScript Compiler

The GWT Java-to-JavaScript compiler translates the Java programming language to the JavaScript programming language. GWT compiler is used to run GWT applications in web mode.

GWT Hosted Web Browser

The GWT Hosted Web Browser lets us run and execute GWT applications in hosted mode, where code runs as Java in the Java Virtual Machine without compiling to JavaScript. To accomplish this, the GWT browser embeds a special browser control (an Internet Explorer control on Windows or a Gecko/Mozilla control on Linux) with hooks into the JVM. This mode make easy develop phase and testing process.

JRE emulation library

GWT contains JavaScript implementations of the most widely used classes in the Java standard class library, including most of the java.lang package classes and a subset of the java.util package classes. The rest of the Java standard library is not supported natively within GWT. For example, packages like java.io do not apply to web applications since they access the network and local file system.

GWT Web UI class library

The GWT web UI class library is a set of custom interfaces and classes that let us create web browser "widgets," like buttons, text boxes, images, and text. This is the core user interface library used to create GWT applications.

5.2.2 SIP

The Session Initiation Protocol (SIP) is an Internet Engineering Task Force (IETF) [21] standard protocol for initiating an interactive user session that

involves multimedia elements such as video, voice, chat, gaming, and virtual reality. SIP is a request-response protocol, dealing with requests from clients and responses from servers. [5], [6]

5.2.3 DVTS

DVTS (Digital Video Transport System) is a software that allows to encapsulate DV (Digital Video) video format from an IEEE Standard 1394 (IEEE1394) interface for transmission over IP networks, resulting in a high quality DV video stream that consumes roughly 30 Mbps of bandwidth.

DVTS works on varying operating systems and connection between PC and DVTS system can be using Firewire standard interface (IEEE1394) [22], [23]. Also DVTS was standardized by the IETF and is Open Source.

At implementation level we can mention some problems detected in the platforms.

Windows version use DirectX to reproduce received video without have any FireWire output device connected to PC. The code has been created with DirectX version 8.1. At present, this version is obsolete and is not available to download from Microsoft home page [24], [25]. The new version (9.0c) has changed libraries used in DVTS code. This issue forces us to use DirectX SDK 8.1 to compile the code.

There is not much documentation of the DVTS Windows version and the existent documents are in Japanese. There is not documentation of other working groups that are working with this version. For the compilation of the code is needed to follow some steps, compiling different libraries. The document that contains these steps also is in Japanese. However, it can do it intuitively.

After compilation process we get multiple errors caused by lack some references in the code. For solving this problem should be studied one to one the different errors (these errors appear in intermediate step, therefore is possible that afterwards we get more errors)

The code is structured in classes, although all comments are in Japanese, which difficult analysis.

Windows version must be modified because it does not accept parameters. This supposes that SIP client cannot configure parameters and establish the videoconference automatically. Therefore, user may do this task losing transparency effect

However DVTS Linux version does not need adaptation and FireWire works correctly to reproduce received video but with some errors in the image and low performance.

At present DVTS integration in Linux version works correctly, is possible to receive as well as to send the video. They are the requirements that the user has to attain:

For video transition, we need dvsend application (included in the DVTS). Dvsend needs that your computer accomplishes with IEEE1394 environments rules. Newest Linux kernels are ready. If not, we must compile DVTS. To compile it is necessary that FireWire libraries (1394) were installed, moreover we need sources code file of the kernel. Finally, it is necessary to mount the device every time it is connected.

For video reproduction in receiving machine, it is necessary to compile an application, which is distributed by the creators of DVTS in a separate pack. For compiling it, we need install DV libraries, as well as the sources code files of the X11.

5.2.4 UltraGrid

UltraGrid is a high definition (HD) video conferencing and distribution system. It is also considered the first system capable of supporting uncompressed gigabit rate high definition video over IP. In fact, an Ultragrid node convert SMTPE 292M high-definition video signals into RTP/UDP/IP packets, which can be distributed across an IP network reaching transmission rates until 1.2Gbps.

Ultragrid is a software OpenSource developed by the ISI EAST [26] with the main goal to stress network launching a flow of video of high quality (HD) without compressing, in order to provoke congestion. This software has double function capture / display and transmitter / receiver, correspondingly. Therefore, it takes video on real-time and packs it in IP datagrams to be sent over network. In reception, it made opposite work, receiving, unpacking and retrieving HD-SDI signal for any device that can understand it.

Some problems detected on testing process are loss of frames for second, caused by low processed capacity or card (it points are pending to demonstrate). For a high definition transmission, we notice how video have some small cuts, which cause a sensation of instability. Theoretical models say that it should send around 900 Mbps, but in practice, throughput is never over 800 Mbps.

5.2.5 Starting Videoconference program

In section 3.1 was commented that open programs from a web browser is complicated by security restrictions. To open programs from web browser is possible with one of the following options:

- Using plugins
- Using Java Web Start [27]
- Modifying windows registry, associating a protocol with a program. [28]

In this moment, there are not any DVTS or Ultragid plugin for web browser, therefore, we should have to implement it.
Java Web Start has some security problems, and is not trivial pack a big application as DVTS o UG in a Java Web Start package.
Last option is only available in Windows, but UG is not available in this platform, and Windows version of DVTS, how is comment on section 5.2.3, does not accept parameters, therefore is impossible to configure it.

To solve this problem, we decide to join client and server machine in only one until we find a solution and application server opens videoconference software.
Nevertheless, this problem may repeat for every videoconference system that appears.

5.3. Project structure

GWT projects are overlaid into Java packages. This structure is fixed by GWT as following manner:

<packageName>.ServiceName
The project root package. It contains module GWT configuration files
<packageName>.ServiceName public
Static resources with public access and placed in server.
<packageName>.ServiceName.client
Client-side source files and subpackages
<packageName>.ServiceName.server
Server-side code and subpackages

packageName is: *net.i2cat.machine*
Service name is: *thinclient*

Client-side code will be translated to JavaScript code by GWT compiled. It implies that in this package we only can use supported classes by GWT. For example, we cannot use threads, hashtables or functionalities of java 5.0. It is a serious limitation and a factor key in implementation phase. This limitations are explained in section 5.4.3.

However, on server we can use all features of java and external libraries. It is an important characteristic because allows us to use SIP in our project. Server-side code is similar than traditional servlet. External libraries used are JainSIP for SIP communication or SAX for XML proposes.

5.4. Communication interface

To communicate from web application to web server, GWT provides a Remote Procedure Call (RPC) mechanism. We define serializable Java classes for requests and responses and GWT automatically serializes the request and deserializes the response from the server.

The server-side code invoked from client is often referred as “service”, so the act of making a remote procedure call is sometimes referred to as “invoking a service”.

To design interface communication between client and service we must to define two java interfaces and one java class, in green in Fig. 5.3. They must have related as shows Fig. 5.3. Orange components are GWT’s class.

GWT defines a naming agreement for this Java files. The synchronous interface must finish in “Service”. Asynchronous interface must named as synchronous attaching at the end “Asynch“ suffix. Finally, implementation of our Service must name as synchronous interface adding at the end “Impl” suffix.

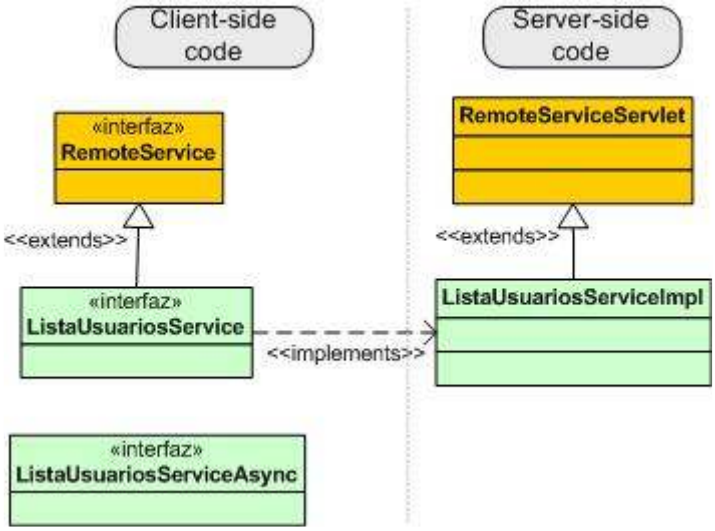

Fig. 5.3 Communication interface

We can see part of code for every class in following tables.

Table 5.1 Synchronous Service Interface

```
public interface ListaUsuariosService extends RemoteService {
        public Usuario getUser(String name);
```

Table 5.2 Asynchronous Service Interface

```
public interface ListaUsuariosServiceAsync {
        public void getUser(String name, AsyncCallback callback);
```

Table 5.3 Service Implementation

```
public class ListaUsuariosServiceImpl extends RemoteServiceServlet
implements ListaUsuariosService{

        public Usuario getUser(String name){
                return ListaServer.getInstance().getUserByName(name);
        }
```

5.4.1 Asynchronous Method Call

The nature of asynchronous method calls requires the caller to pass in a callback object that can be notified when an asynchronous call completes, since by definition the caller cannot be blocked until the call completes. For the same reason, asynchronous methods do not have return types; they must always return void. After an asynchronous call is made, all communication back to the caller is via the passed-in callback object.

An analogy of these types of calls can be explained by the following example. When we request something on a supermarket and we must attach a box. The response is delivered packed inside our “intelligent” box to home. This box has some instructions for what to do with content. In addition, we are not waiting to receive the response and handle it; we can attend to other things.

This box is the AsyncCallback shows on Table 5.4

Table 5.4 Asynchronous method call

```
listaUsuariosService.getUser(name, new AsyncCallback(){
        public void onFailure(Throwable caught) {
                // Failure code
        }

        public void onSuccess(Object result) {
                // On Success code. Result may cast. For example:
                //        Usuario us = (Usuario) result;
});
```

5.4.2 Serialization types

At the beginning of section 5.4 was explained that is possible to use Java class for requests and responses and GWT serializes them to send across network.

Not all Java class can be serialized; these classes must conform to certain restrictions.

A type is serializable and can be used in a service interface if it

- is primitive, such as char, byte, short, int, long, boolean, float, or double;
- is String, Date, or a primitive wrapper such as Character, Byte, Short, Integer, Long, Boolean, Float, or Double;
- is an array of serializable types (including other serializable arrays);
- is a serializable user-defined class; or
- has at least one serializable subclass

And user-defined class is serializable if it is assignable to IsSerializable interface. This interface is a “GWT test“, which certifies if a class can be serialized or do not. Only detail explained by GWT creators is that a class is serializable if all non-transient fields are themselves serializable. On implementing phase some restrictions have been founded, as example, all classes must contain default constructor (without parameters).

5.4.3 Limitations on client-side

GWT supports only a small subset of the classes available in the Java 2 Standard and Enterprise Edition libraries, as these libraries are quite large and rely on functionality that is unavailable within web browsers.

Main limitations in this sense can be noticed at the beginning because no habit or type of data not supported, and common methods not implemented in GWT.

Note that these libraries have been improved during project developing. Unsupported functionalities, which have been solved with personal implementations, on following releases were resolved, supposing that delivered hours in lost time

In this sense, mention that since September until February has appeared three releases.

5.4.4 User interface toolkit

GWT user interface classes are similar those in existing UI frameworks such as Swing and SWT except that the widgets are rendered using dynamically-created HTML rather than pixel-oriented graphics.

GWT provides several classes to develop “widgets” as buttons, tables, text box and layouts to allocate this widgets, also provides mechanism to manage browser events.

CHAPTER 6. PLANNING AND COST ESTIMATION

This chapter describes how has been planned this project, how many time has been spent in every task and a cost estimation.

6.1. Planning

Project is divided in 5 phases

- Previous research
- Design
- Implementation
- Integration and Testing
- Documentation

Previous research

In this phase has been done a research of current state-of-the-art of technology in order to identify which technologies are useful for our proposes. It supposes to do a study and evaluation for each candidate to determine the suitable solution. Once has been chosen which tool to use, it is needed a learning curve.

Design

In this phase has been closed down all features. Application architecture, and how design every piece have discussed in this stage.

Implementation

Implementation consist of developing designed application. First step is centred to get a prototype with basic functionalities. Second stage tries to complete the prototype implementing all features, delivering the first release. Next steps consist in improve failure tolerance, application look. Many steps need a testing process to approve the prototype.

Integration and Testing

Testing task was done in many intermediate stages, therefore is a discontinuous task. Implant our releases on current environment and ensure that its behaviour is correct is located in this phase too.

Documentation

Writing all documents necessary to close project. User manual, installation manual and project report.

Table 6.1 shows hours took by each task. Total dedication is around 500 hours.

Table 6.1 Task dedication

Phase	Task	Description	Hours
Previous research	First contact	Posing the problem and analysis possible solutions in a couple of meeting with project tutor. These meetings conclude with the propose of a possible solution	8
Previous research	Toolkit research	Research of different AJAX toolkits and test them	15
Previous research	GWT learning	Getting experience with GWT toolkit, investing all features.	35
Design	Features required	Defining all functionalities that our project need to implement	25
Design	SIP and Presence Agent Interface design	Meeting with SIP developer to arrange a communication interface.	8
Design	Architecture definition	Have been defined main characteristics of application architecture	12
Design	SIP features	Design news features of SIP Client.	8
Design	Presence Agent	Design Presence Agent interface, communication protocol.	15
Implementation/Previous Research	First AJAX application	First service on AJAX which share data between server and client	20
Implementation	First prototype	First prototype which include and static list. This list is loaded from server and can be modified by client.	60
Implementation / Testing	Persistent connection	Implement mechanism to keep an active connection to communication from server to client	15
Implementation / Testing	SIP Client	Equip our prototype with a SIP Stack.	14
Implementation / Testing	Presence Service	Interact with Presence Agent. Get a dynamic list, notify our changes and integration with persistent connection	10
Implementation	Look of application	CSS design and several proofs	8
Implementation/Testing	Add multimedia layer	DVTS installation, and integration in application	17
Implementation	Graphic pannels	Test all panels available and design all "web pages"	50
Testing	Global test	Test all possibilities, solving several errors	40
Integration	Migration to other platform	Several issues to migrate from Windows system to Linux	8
Integration	Migration to stable scenario	Migration to an stable scenario for demo propose.	4
Documentation	Report and presentation	Report project, presentation.	110

6.2. Cost Estimation

In order to make a cost estimation of the project, some aspects must be considered.

Time devoted

Planning in section 6.1 details all task and how many time has been spend every tasks. This project must be realized in 5 months approximately, which supposes time for research, design, implementation and testing of the application.

Achieved objectives

The cost estimated can depend on final results. A project can suppose a knowledge base or small prototypes that allow in future to work in ambitious project. The worst extreme should be when all time spent has produced zero results.

Risks

Initial requirement of the project was research some toolkits to achieve manage a SIP client. We decide for a newbie product, which has appeared 4 month before to start this project. This kind of products may death in few months and become abandoned projects without documentation or support

Other risk was to choose a toolkit, which limits our purposes; in this case, we get right.

Staff

A single student of Telecommunications Engineering realizing the final thesis of the M.S. degree in Telematic Engineering. The student does not receive any salary during the project duration.

Tools and equipment

Developing and testing phase requires two client machines on Linux distribution and another machine for SIP server proposes. Additionally we need a Fast Ethernet switch, replaced by a Gigabit Switch on November to connect all PCs of our scenario.

All used tools are free licence, for example develop environment tool used was Eclipse Project.

The University has assumed all cost related with project, therefore in our cost estimation will be a difficult data to predict. Comments that software and hardware are shared with other projects.

Economical Result

Know all costs associated to project is an impossible task. We have to keep satisfied with a relevant parameter. A representative aspect may be staff cost, which has been calculated on Table 6.2. The University determines minimum cost per hour in 7 €/hr

Table 6.2 Cost estimation of the project

Hours per day (hr)	5
Total amount of days	100
Cost per hour (€/hr)	7
TOTAL	3500€

CHAPTER 7. CONCLUSIONS

This chapter explains the archived objectives, tasks that are able to do in future in this project, environment impact and finally personal conclusions.

7.1. Achieved objectives

We get a web interface that is according to requirements.

- ✓ Web access
- ✓ Authentication
- ✓ Edit your profile
- ✓ Contact list
- ✓ Videoconference management

We get an application capable to manage a SIP stack with functionalities

- ✓ Establish / Finish a conference
- ✓ Invite to a conference
- ✓ Receive invitation and manage it
- ✓ Communication with Presence agent, managing contact list information

We get an application capable to

- ✓ Run a videoconference software
- ✓ Configure a videoconference software
- ✓ Close a videoconference software

Therefore, we can announce that we have obtained a web interface, which manages a SIP stack and can interact with other standards SIP clients and is capable to initiate videoconferences and manage it.

On other hand, a first release of presence agent has been developed successfully, which allows basic functions and it is easily improved.

Finally, we get a tool for carrying out high-definition videoconference point to point. This operation is done abstracting user of signalling task and becoming a transparently process.

All integrated in the same application and compatible with other modules of i2Cat Foundation.

7.2. Improvements and future work

Web interface can be improved as following:

- To be able to start videoconference software in client machine. Research lines are:
  - Without previous software installed:
    - Java Web Start
    - Embedding a plugin as Flash, Videolan or develop a DVTS plugin

    - Software preinstalled
        - Proprietary protocols defined on system register that allow run an application with parameters.
- Improve look&feel and design application.
- To become most reliable and robust on communication failures, especially from server to client communication.

Application can be improved
- Allowing multiconference
- Supporting pay-per-use accounts

7.3. Environment impact

Nowadays, in a globalized world, working groups are distributed around the world. Several firms have some local office around a country or continent and every month high-level managers travel for meeting on headquarters. It is only an example where videoconference can be useful to save time, money and can reduce pollution.

Easy videoconference systems with high quality can suppose useful for working or studying long distance. Two actual cases are following. Every Friday, Ed Seidel, director of the Center for Computation & Technology (CCT) at Louisiana State University, gives a lesson to PhD Czechs students. This lesson is transmitted in real-time across network. Another example; two professors from Technical University of Catalonia in Barcelona give PhD lessons to student of University Carlos III in Madrid, transmitted also across network.

Negative impact may be needs of high amount of bandwidth that are solved with constructing news emplacement for optical fiber, satellite communications, aerial constructions which break skyline.

7.4. Personal conclusions

This project has involved me to know i2Cat Foundation, giving an opportunity to work in their installations and know research areas unexplored in my studies. I began this project with poor knowledge about SIP, videoconference, high-definition and rich internet applications and at the end, I am able to manage all them with acceptable agility.
In other hand, this project has been a great opportunity for integrating to a workgroup and develop a project with several blocks. I think that working on group is an important experience.

I want to denote too that this project has given me the chance to practice my forgotten English. I notice that my level is insufficient and that I need to take classes.

CHAPTER 8. REFERENCES

[1] DVTS, DV Stream on IEEE1394 Encapsulated into IP. *URL:* http://www.sfc.wide.ad.jp/DVTS/

[2] C. S. Perkins and L. Gharai. UltraGrid: a high definition collaboratory, Nov. 2002. *URL:* http://ultragrid.east.isi.edu/.

[3] AJAX Introduction *URL:* http://www.w3schools.com/ajax/

[4] M. Hurtado, A. Oller, and J. Alcober, "The SIP-CMI Platform- An Open Testbed for Advanced Integrated Continuous Media Services," TridentCom 2006.

[5] M. Handley, H. Schulzrinne, E. Schooler and J. Rosenberg “SIP: Session Initiation Protocol” IETF RFC 2543, March 1999. Obsolete and updated by [6]

[6] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," IETF RFC 3261, June 2002.

[7] I2Cat Foundation Homepage *URL:* http://www.i2cat.net/

[8] SER, SIP express router. *URL:* http://www.openser.org/

[9] Eckel, B., *“Thinking in Patterns with Java”*. Chapter 12 *“System decoupling“* Electronic Book in development. URL: http://mindview.net/Books/TIPatterns/

[10] Manufacture of hardware components. LongShine Iberia *URL:* http://www.longshine.es/

[11] Java Technology Homepage. *URL:* http://java.sun.com/

[12] Eclipse - an open development platform *URL:* http://www.eclipse.org/

[13] Log4J Project *URL:* http://logging.apache.org/log4j/

[14] SAX Project *URL:* http://www.saxproject.org/

[15] Apache Ant Project *URL:* http://ant.apache.org/

[16] Dojo, the Javascript Toolkit *URL:* http://dojotoolkit.org/

[17] MochiKit - A lightweight Javascript library *URL:* http://mochikit.com/

[18] Google Web Toolkit - Build AJAX apps in the Java language *URL:* http://code.google.com/webtoolkit/

[19] Google Maps *URL:* http://maps.google.es/

[20] GMail *URL:* http://www.google.com

[21] IETF Home Page *URL:* http://www.ietf.org/

[22] The world's leading professional association for the advancement of technology. IEEE Homepage. *URL:* http://www.ieee.org/

[23] IEEE Standards *URL:* http://www.ieee.org/web/standards/home/index.html

[24] Microsoft Corporation *URL:* http://www.microsoft.com/

[25] Microsoft DirectX: Home Page *URL:* http://www.microsoft.com/directx/

[26] Information Sciences Institute East *URL:* http://www.east.isi.edu/

[27] Java Web Start Technology
URL: http://java.sun.com/products/javawebstart/

[28] Registering an Application to a URL Protocol.
http://msdn.microsoft.com/library/default.asp?url=/workshop/networking/pluggable/overview/appendix_a.asp

CHAPTER 9. ACRONYMS

AJAX	Asynchronous JavaScript and XML
AMD	Advanced Micro Devices
CSS	Cascading Style Sheets
CVS	Concurrent Versioning System
DV	Digital Video
DVTS	Digital Video Transport System
GUI	Graphical User Interface
GWT	Google Web Toolkit
HD	high definition
HTML	Hypertext Markup Language
HTTP	Hyper Text Transfer Protocol
i2Cat	Internet 2 a Catalunya
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
JDK	Java Development Kit
JRE	Java Runtime Environment
PC	Personal Computer
RAM	Random-Access Memory
RFC	Request for Comments
RPC	Remote Procedure Call
RTP	Real-time transport protocol
SAX	Simple API for XML
SDK	Software Development Kit
SER	SIP Express Router
SIP	Session Initiation Protocol
SIP-CMI	SIP-based Continuous Media Integration
SMTPE	Society of Motion Picture and Television Engineers
UDP	User Datagram Protocol
UG	Ultragrid
UI	User Interface
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language

ANNEX A. SCREENSHOTS

The user must identify at startup. Fig. A.1 shows form that must be fulfilled.

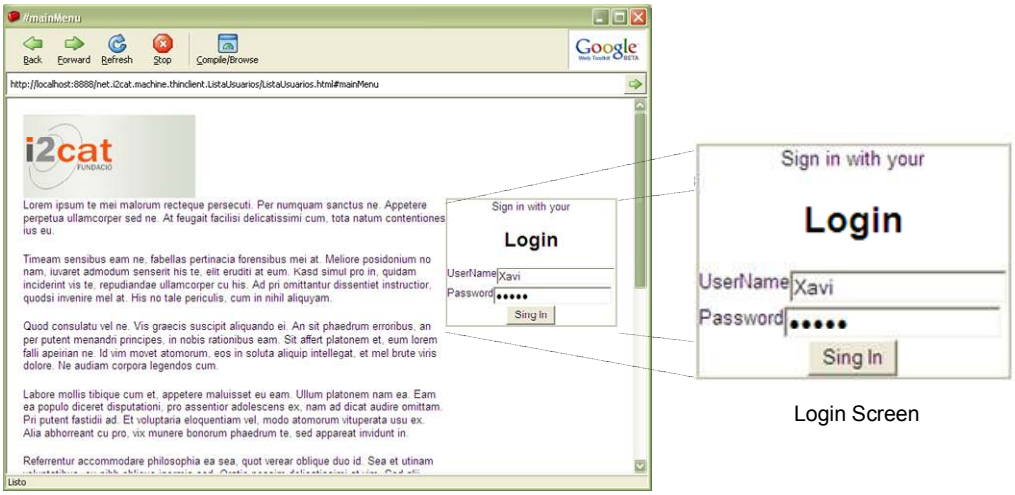

Fig. A.1 Login Screen (Web Interface)

Fig. A.2 Login Screen (Swing Interface)

Once time user has been logged, he may modify profile, changing Multimedia Capabilities (Fig. A.3). Left panel shows menu list and Contact List.

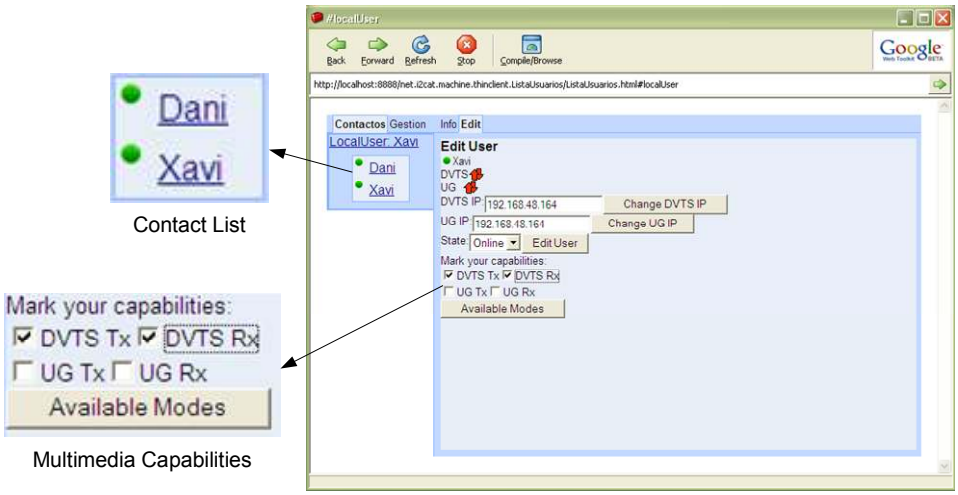

Fig. A.3 Local user profile modification (Web Interface)

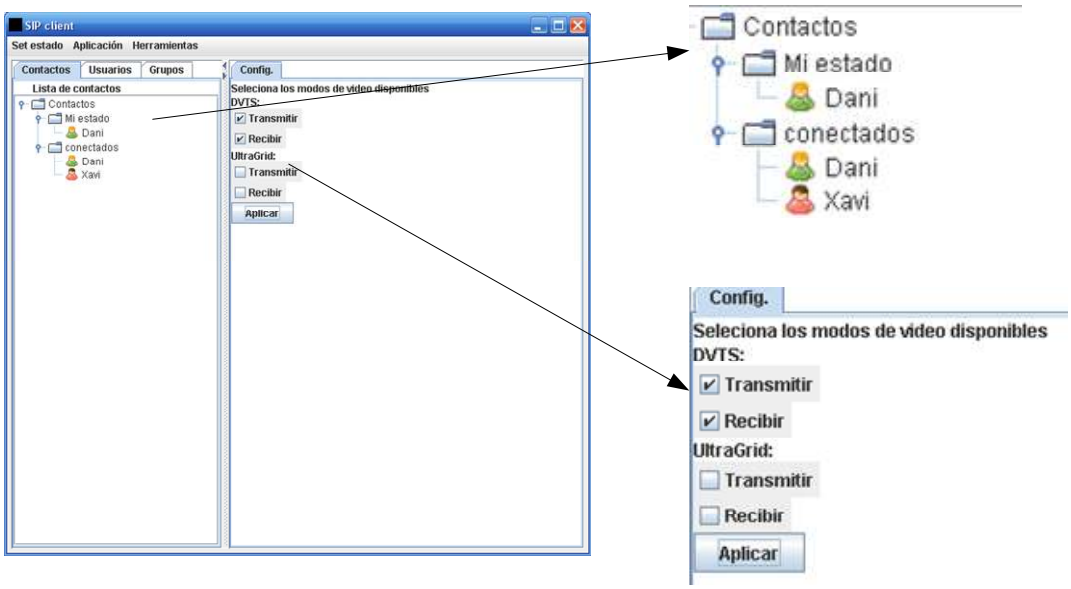

Fig. A.4 Local user profile modification (Web Interface)

Clicking over a contact appears contact profile (Fig A.5). This panel contains a graphic representation of his multimedia capacities and his state.
For calling to a contact, are enabled all modes compatibles between contact and application user. In Fig A.5 are enabled all DVTS modes and UG are disabled because we don not support it.

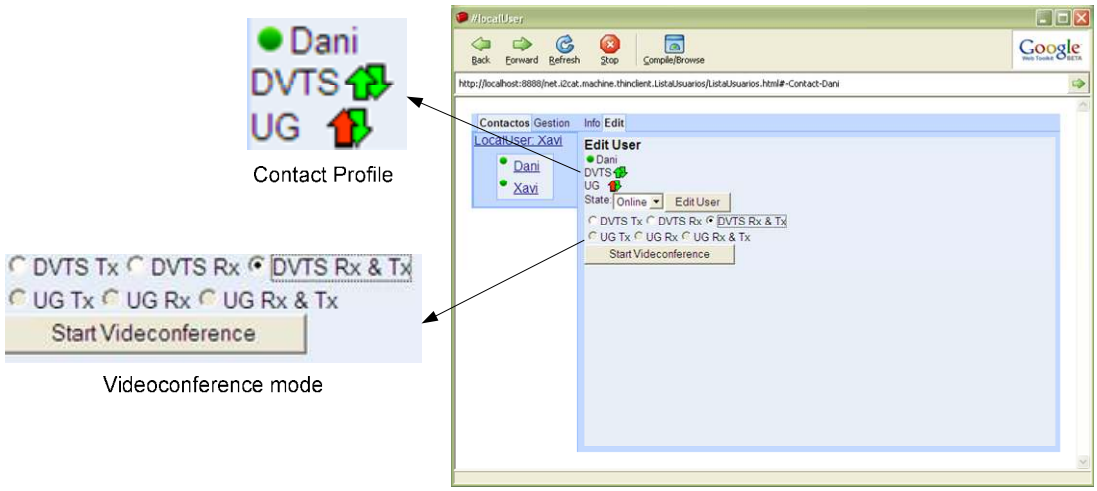

Fig. A.5 Contact profile (Web Interface)

When we receive an incoming call appears a popup which allows us Accept or Reject this videoconference invitation. (Fig. A.6)

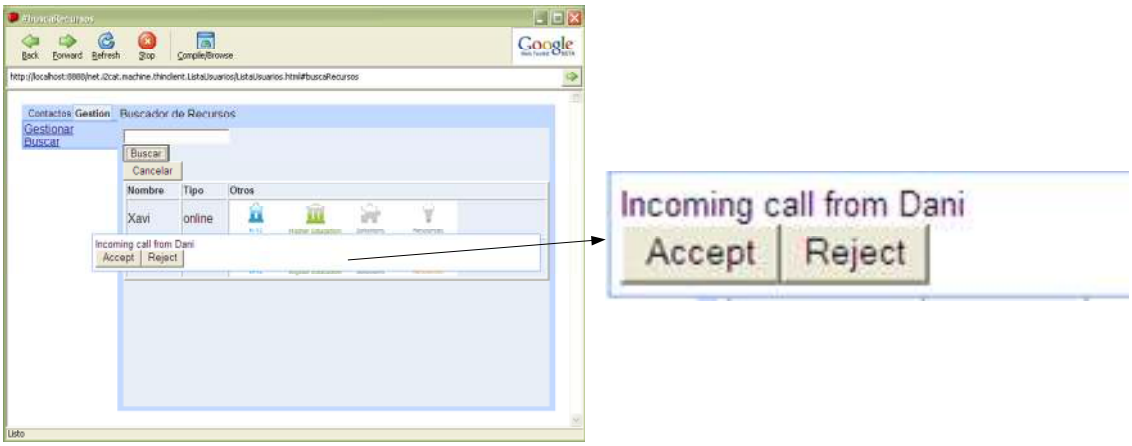

Fig. A.6 Incoming call (Web Interface)

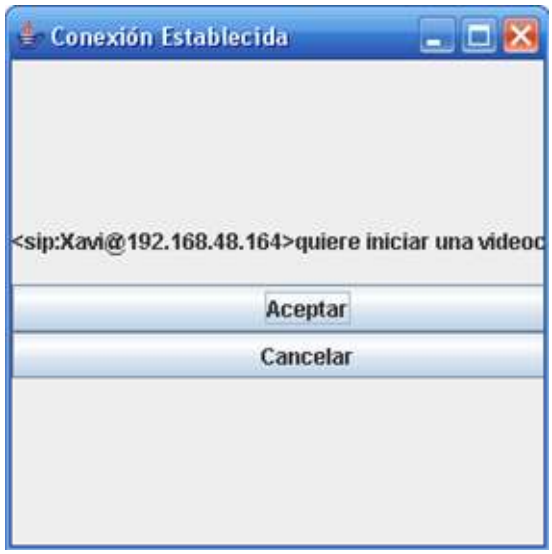

Fig. A.6 Incoming call (Swing Interface)

ANNEX B. USER GUIDE

This guide explains how to execute developed application. This is not an installation manual.

Minimum requirement

- Pentium IV 2,4 Ghz
- 512Mb RAM
- Fast Ethernet network interface card
- 40 GB hardisk
- Ubuntu Linux 6.06 LTS
- DV Camera

Previous task in Ubuntu Linux SO:

- Installation of JSDK 1.4 or upper of SUN
- Installation of Apache Ant 1.6 or upper
- Installation de Eclipse IDE
- Installation of UG and/or DVTS (www.sfc.wide.ad.jp/DVTS)

1st Download CVS module

CVS Client configuration (Fig. B.1):

Host: broadband6.upc.es
Repository path: /home/cvs/
Connection type: extssh

Project is located in “/machine/ajax”

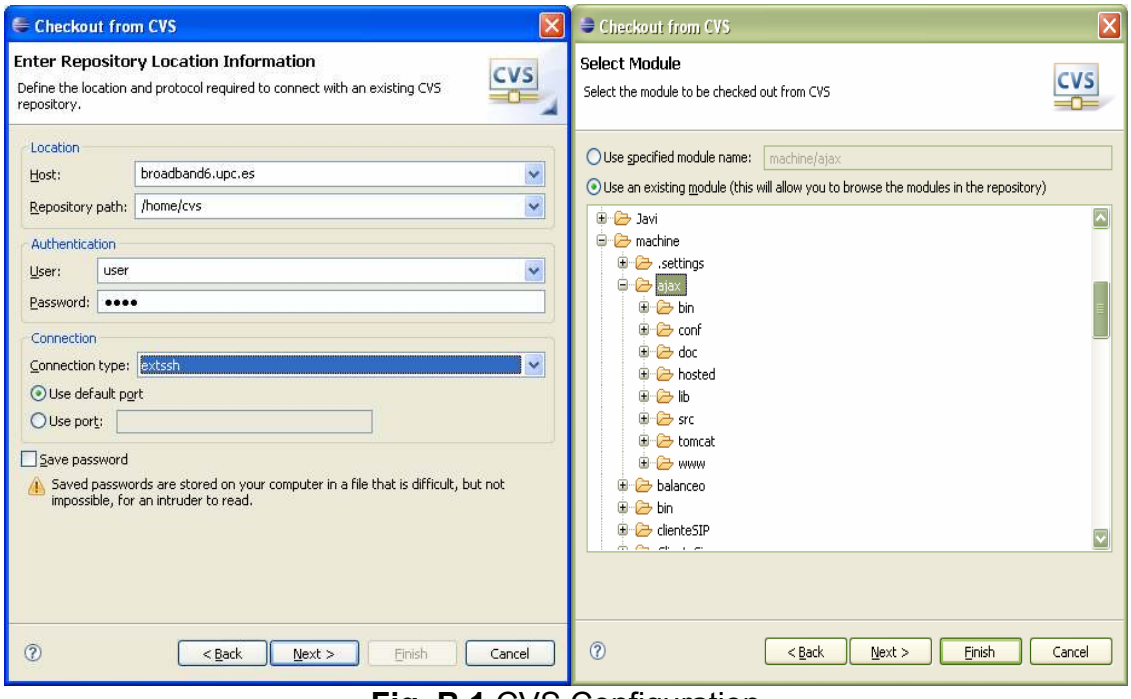

Fig. B.1 CVS Configuration

2nd ClienteSIP file

"ClienteSIP.propieties" must be configured with as following:

ip=192.168.48.164	* IP of local machine
port=5060	* Application port to use
proto=tcp	
domain=192.168.48.83	* SER IP.
proxy-registrar=192.168.48.83:5060	* SER IP and PORT
UGRXpath=./uv	* Ultragrid Path
UGRXmod1=-d	
UGRXmod2=xena	
UGRXmod3=-c	
UGRXmod4=1	
UGRXmod5=-b	
UGRXmod6=8	
UGRXIP=127.0.0.1	* Machine where reproduce UG video received.
UGTXpath=./uv	* Ultragrid Path
UGTXmod1=-t	
UGTXmod2=hdtv	
UGTXmod3=-c	
UGTXmod4=1	
UGTXmod5=-b	
UGTXmod6=8	
DVTSRXpath=xdvshow	* Xdvshow path
DVTSTXpath=dvsend	* Dvsend path
DVTSTXmod1=-h	
#DVTSTXIP=192.168.48.137	* IP remote to send video. (Obsolete)

3rd .properties files

Copy all.properties files to "bin" folder.

4th Execute program

- Click on Run>> Run… (Fig. B.2)
- Select ListaUsuarios on left menu.
- Click on Run button

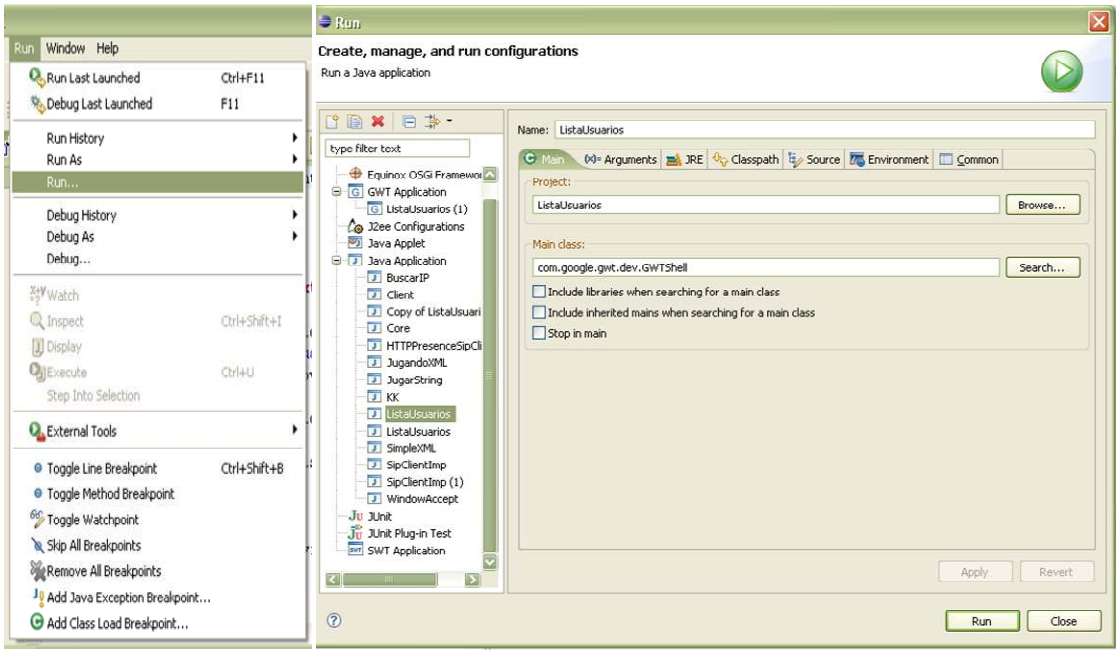

Fig. B.2 Eclipse configuration

5th Application Working

Two windows open, a control windows and graphic interface. Enjoy it!