Escola Politècnica Superior
de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL DE FI DE CARRERA

TÍTOL DEL TFC: Serveis de control i consum de recursos multimèdia

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Telemàtica

AUTOR: Oriol Solé Molina

DIRECTOR: Antoni Oller Arcas

DATA: 13 de juliol de 2007

Títol: Serveis de control i consum de recursos multimèdia

Autor: Oriol Solé Molina

Director: Antoni Oller Arcas

Data: 13 de juliol de 2007

Resum

En aquest document es descriu el disseny i desenvolupament de serveis telemàtics per al servei i control de sessions multimèdia. S'exposa tant el disseny de tots els elements que conformen el sistema com la seva implementació.

L'objectiu principal és, a més de servir i controlar qualsevol sessió multimèdia, permetre que tant el client com la base de dades del sistema es trobin en llocs diferents de la xarxa, sense necessitat de conèixer la seva localització. Per tant, la mobilitat d'alguns dels elements, és un dels punts clau que es tracten en aquest projecte.
Es contempla, també, la possibilitat d'incorporar algun dels elements descrits en un projecte de la fundació i2Cat en procés de desenvolupament: el projecte *Machine*.

S'ha realitzat l'estudi de dos protocols de senyalització i control de comunicacions: SIP i RTSP. En l'estudi d'aquests protocols s'ha vist com eren de versàtils i les possibilitats que oferien. De la capacitat de localització d'usuaris de SIP i del control de sessions multimèdia d'RTSP, va sorgir la idea de dissenyar i desenvolupar tot el sistema.

El resultat final ha estat una infraestructura de servei de Video sota Demanda, LiveStreaming amb capacitat de control de les sessions multimèdia on el servidor central d'informació es pot trobar en qualsevol punt de la xarxa sense necessitat de que el client conegui la seva localització.

Title: Control and consume services of multimedia resources

Author: Oriol Solé Molina

Director: Antoni Oller Arcas

Date: July, 13th 2007

Overview

In this document is described the design and development of telematic services to serve and control multimedia sessions. All the system elements design and their implementations are presented.

The main objective is, also to serve and control any multimedia session, to let the client and the system database be in any place without the need to know their network location. So the elements mobility is one of the main goals dealed in this project.
Also is contempled the possibility to incorporate some of the elements described in an i2Cat foundation project: the *Machine* project.

It has done studies on two signalin and control protocols: SIP and RTSP. In this protocol study it has seen how versatile they are and wich possibillities they offer. The system design and development were born from the SIP user location and the RTSP multimedia control session capabilities.

The final result has been a Video on Demand and LiveStreaming service infrastructure with multimedia control session capacity where the central information server can be in any place without the need to know which his network location is.

Agraeixo a totes aquelles persones que han donat suport a la realització d'aquest treball: al tutor Toni Oller per la seva paciència i dedicació, a en Sergi Laencina per la informació facilitada sobre la Passarel·la SIP-RTSP del projecte *Machine*. A tots els companys de la fundació i2Cat per l'ajuda i suport facilitats. Finalment al meu company de trifulques, Guillermo Enero, per les bones estones i per la compartició de coneixements Java.

ÍNDEX

INTRODUCCIÓ

Cada cop més apareixen serveis de vídeo sota demanda i de flux en viu com els que ofereixen webs com YouTube o emissores de radio i televisió com iCatFM i Televisió de Catalunya.

El gran inconvenient de tots aquests serveis es troba quan es vol disposar d'un d'aquests recursos però realment no se sap on acudir. Webs com YouTube ofereixen un cercador per a que trobar el que es busca no sigui tant feixuc. Però l'inconvenient de YouTube és que aquest només cerca els seus propis recursos.

Una de les solucions és la creació d'una base de dades en la que s'emmagatzemi tota la informació referent a la localització de tots els recursos multimèdia que es desitgin consumir en un futur. Així sempre es podrà saber on es troba un vídeo o emissora de ràdio preferits.

Però desar la informació d'un recurs no permet el seu consum. Caldrà doncs un element que així ho permeti, com un reproductor de vídeo capaç de rebre fluxos multimèdia a través de la xarxa.

Tota aquesta problemàtica és el que s'intenta solucionar amb la realització d'aquest treball. A més d'oferir la possibilitat de compartir tota la informació amb altres usuaris sense que aquests s'hagin de preocupar de la localització de la base de dades.

Inicialment s'ha estudiat les possibilitats que oferien protocols de control i senyalització de comunicacions com SIP i RTSP. Fruit d'aquest estudi es va veure que el protocol de senyalització SIP permet que els usuaris es puguin comunicar entre ells sense preocupar-se de la seva situació en la xarxa. També es va veure que el protocol de control de sessions multimèdia RTSP permet el diàleg necessari per establir una sessió multimèdia amb el servidor d'un recurs per tal de rebre el seu flux.

Al finalitzar l'estudi de SIP i RTSP es van definir uns objectius a assolir en acabar el projecte. Donar servei de Vídeo sota Demanda i LiveStreaming, permetre la mobilitat dels elements participants del sistema i la possible integració en el projecte *Machine* desenvolupat per i2Cat, són els objectius que es volen assolir al finalitzar el desenvolupament d'aquest treball.

L'estudi i desenvolupament d'aquest treball ha seguit un ordre cronològic com queda palès en la estructura d'aquest document.

Inicialment es va realitzar, com s'ha comentat, l'estudi dels protocols SIP i RTSP, tal i com s'exposa en el segon capítol.

A continuació es va realitzar un estudi dels elements necessaris per a la implementació dels dos protocols.

Seguidament, en el quart capítol, s'exposa el disseny de cada element que conforma el sistema.

Finalment es descriu quina ha estat la implementació, basada en el disseny inicial, de cada element.
S'inclouen, també, capítols en els que s'exposa el pla de treball seguit per al desenvolupament del projecte, i les conclusions a les que s'ha arribat després de la realització del treball.

Durant la lectura d'aquest document es poden trobar referències a peu de pàgina on insten al lector a acudir als annexos per tal d'aprofundir coneixements en aquells conceptes que així s'indiquen.

CAPÍTOL 1. ESPECIFICACIÓ

En tot projecte de desenvolupament cal definir i especificar quines fites es volen assolir. Cal, doncs, especificar quins són els objectius que es volen cobrir amb la realització del projecte. Al llarg d'aquest capítol es descriuen aquests objectius (apartat 1.1), serveis multimèdia considerats (Vídeo sota demanda i LiveStreaming, apartats 1.2 i 1.3) i la possibilitat d'incorporar el sistema en un projecte existent (Gateway SIP-RTSP del projecte Machine, apartat 1.4).

1.1. Objectius

Els principals objectius del treball que es descriu en aquest document són quatre, tots ells relacionats amb la prestació de serveis telemàtics:

- Realització d'un prototipus de gestió i consum de recursos multimèdia (Video sota Demanda)
- Incorporació d'un servei de ràdio en directe. LiveStreaming.
- Permetre la mobilitat dels elements participants.
- Permetre la integració del servei en el mòdul Gateway SIP-RTSP del projecte *Machine* de i2Cat [23]

El diagrama de casos d'ús de tot el servei és el següent:

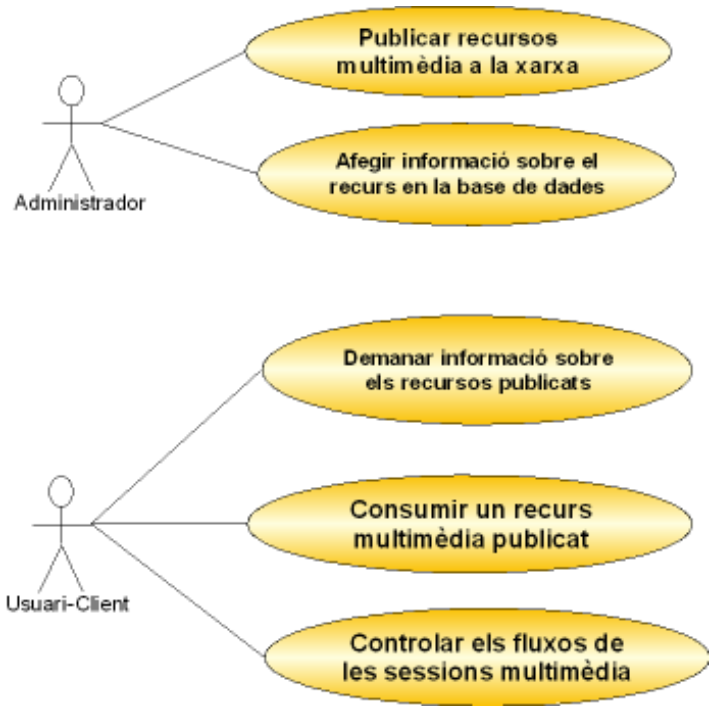

Fig.1.1. Diagrama de casos d'ús del servei a desenvolupar

A continuació es descriuen, en forma de taula, les funcions de cada actor:

Taula 1.1. Descripció de les funcions de cada actor

Funció	Publicar recursos multimèdia a la xarxa
Descripció	Donar d'alta un recurs multimèdia per permetre als usuaris consumir-lo
Actor	Administrador del sistema

Funció	Afegir a la base de dades la informació referent als recursos publicats
Descripció	Gestionar la base de dades per a que aquesta contingui tota la informació referent als recursos publicats.
Actor	Administrador del sistema

Funció	Demanar informació sobre els recursos publicats
Descripció	Demana la llista de recursos disponibles juntament amb informació relacionada com el tipus de recurs: àudio, vídeo o LiveStreaming
Actor	Usuari del sistema

Funció	Consumir un recurs multimèdia
Descripció	Rebre el flux multimèdia relacionat amb el recurs sol·licitat
Actor	Usuari del sistema

Funció	Controlar els fluxos de les sessions multimèdia
Descripció	Controlar el moment que es es dessitja aturar momentàniament i reprendre la recepció dels fluxos multimèdia.
Actor	Usuari del sistema

1.2. Vídeo sota Demanda. VoD.

El servei de Vídeo sota Demanda (en anglés Video on Demand o VoD) és un servei que permet a un usuari consumir un recurs multimèdia de manera personalitzada. L'usuari pot iniciar, aturar i reprendre la reproducció quan ho desitgi.

Serveis de VoD es poden trobar en webs com YouTube[1], on es permet visualitzar continguts multimèdia creats per altres usuaris; en serveis com el *3alaCarta*[2] de Televisió de Catalunya, on es permet visualitzar els continguts de programes ja emesos per aquesta televisió.

En el context del projecte, el servei de Vídeo sota Demanada permetrà als usuaris consumir en el lloc i instant desitjat els recursos multimèdia presents en el sistema.

1.3. LiveStreaming

El concepte de LiveStreaming o flux en viu, fa referència a tot servei d'entrega de fluxos multimèdia procedents d'una font on el temps entre la creació i publicació del flux és de l'ordre dels segons.

Serveis de LiveStreaming són els que ofereixen gran part d'emissores de ràdio i televisió com Catalunya Ràdio[3], Ràdio Flaixbac[4] i Televisió de Catalunya[5].
Aquests serveis són els que permeten visualitzar o escoltar en directe els programes emesos pels diferents mitjans de comunicació audiovisual en indrets on no tenen cobertura per radio freqüència.

El servei de LiveStreaming s'oferirà als usuaris del sistema que es vol implementar a través de la implantació d'un servei de ràdio musical.

[1] YouTube Inc., www.youtube.com
[2] 3alaCarta: servei de vídeo a la carta de Televisió de Catalunya, www.tv3.cat/3alacarta
[3] Catalunya Radio, S.A. www.catradio.cat
[4] Ràdio Flaixbac, S.A. www.radioflaixbac.cat
[5] Televisió de Catalunya S.A. www.tvc.cat

1.4. Passarel·la o Gateway SIP-RTSP

El Gateway SIP-RTSP és un dels elements presents en la implementació del Paquet de Treball de Videoconferència del projecte Machine.
El projecte Machine és un projecte gestionat per la fundació i2Cat i desenvolupat per les universitats Politècnica de Catalunya, Pompeu Fabra i Ramon Llul. Aquest projecte està dividit en 4 paquets de treball i, en un d'ells, el destinat a videoconferència, és on s'inclourà part de la implementació desenvolupada en aquest treball.

La funció del Gateway SIP-RTSP és la de convertir missatges de control de flux multimèdia que es troben dins de missatges SIP en missatges RTSP. D'aquesta manera poder controlar i consumir recursos multimèdia a través de trucades IP.

El diagrama de casos d'ús és el següent:

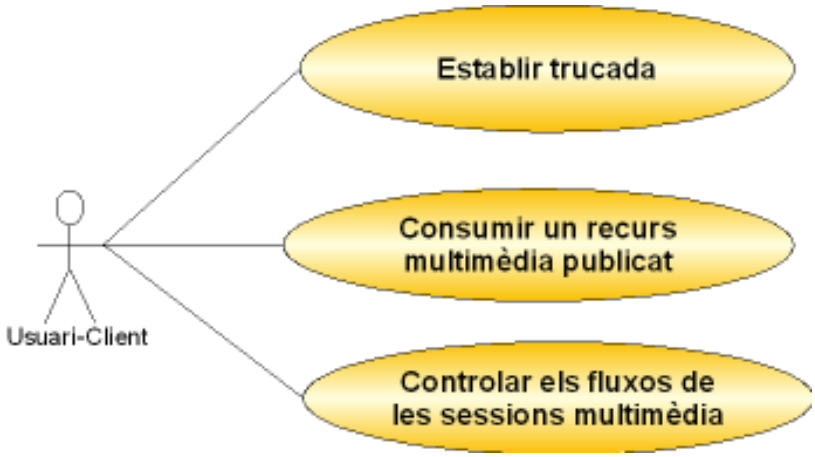

Fig.1.2. Diagrama de casos d'ús usuari de la passarel·la SIP-RTSP

A continuació es presenta, en forma de taules, la descripció de cada funció:

Taula 1.2. Descripció de les funcions de l'usuari.

Funció	Establir trucada
Descripció	Realitzar una trucada a través de la xarxa IP
Actor	Usuari del sistema

Funció	Consumir un recurs multimèdia publicat
Descripció	Rebre el flux multimèdia relacionat amb el recurs sol·licitat
Actor	Usuari del sistema

Funció	Controlar els fluxos de les sessions multimèdia
Descripció	Controlar el moment que es es dessitja aturar momentàniament i reprendre la recepció dels fluxos multimèdia.
Actor	Usuari del sistema

Com es pot observar és molt similar al diagrama de casos d'ús del servei que es desenvolupa en aquest treball (**Fig.1.1**). Per aquest motiu el treball descrit en aquest document dóna suport al desenvolupament d'aquest mòdul.

A continuació es presenta un esquema dels elements participants en el sistema de traducció SIP – RTSP:

Fig. 1.3. Esquemàtic del Gateway SIP-RTSP

El disseny i posterior implementació en aquest treball d'un mòdul de control de sessions RTSP ajudarà en el desenvolupament del Paquet de Treball de Videoconferència del projecte Machine.

CAPÍTOL 2. CONCEPTES BÀSICS

En la implementació del projecte es farà ús dels protocols de senyalització i control SIP i RTSP.
Per a la correcta comprensió de les seves funcionalitats, es va realitzar un estudi d'ambdós protocols. En aquest capítol es presenta una síntesi d'aquest estudi.

2.1. Introducció al Protocol SIP

SIP (Session Initiation Protocol) és un protocol de senyalització definit en l'RFC 3261 de la IETF [1] i és usat en la inicialització, modificació i finalització de sessions multimèdia com per exemple una trucada entre dos punts d'una xarxa.

Per permetre l'establiment, modificació i finalització de sessions multimèdia, SIP presenta les següents característiques:

- Permet la localització d'usuaris. Aquesta característica fa possible la mobilitat dels usuaris, ja que només mantenint un identificador els usuaris són totalment localitzables sigui quina sigui la seva situació dins la xarxa.
- També permet determinar els paràmetres multimèdia que s'utilitzaran en la sessió. Això ho fa fent ús d'un altre protocol anomenat SDP[6].
- Permet també establir paràmetres de la sessió entre els nodes participants, com també permet la modificació d'aquests i la invocació de serveis que puguin oferir els diversos participants.

2.1.1. Capçaleres i Missatges SIP

Excepte algunes diferències, la sintaxi de molts dels missatges i capçaleres SIP són molt similars a les de HTTP[7]. Per posar un exemple, el missatge "404" en SIP indica que una adreça no ha estat trobada, de la mateixa manera que ho indica HTTP.

A continuació es mostra un exemple de missatge SIP seguida d'una breu explicació de cada capçalera:

[6] Session Description Protocol, protocol destinat a la descripció de paràmetres d'inicialització de fluxos multimèdia.
[7] HyperText Transfer Protocol, protocol utilitzat en la transacció de dades de la web.

```
INVITE sip:user2@sip.cat SIP/2.0
Via: SIP/2.0/UDP ser.sip.cat;branch=z9bG6bK655atdhds
Max-Forwards: 70
To: Usuari2 <sip:user2@sip.cat>
From: Usuari1 <sip:user1@sip.cat>;tag=1028601784
Call-ID: a80c4c65e66700@sip.cat
CSeq: 2 INVITE
Contact: <sip:user1@sip.cat>
Content-Type: application/sdp
Content-Length: 142
```

Fig. 2.1. Missatge SIP INVITE

Com es pot observar en la primera línia, aquest es tracta d'un missatge **INVITE** (més endavant es descriuen els tipus de missatges més comuns de SIP) generat per l'usuari SIP user2@sip.cat i la versió del protocol utilitzada és la 2.0.

A la següent línia es troba la capçalera **Via**, que conté l'adreça a la qual l'usuari generador del missatge espera rebre les respostes a aquest missatge. Cada enrutador per on passi el missatge també haurà d'incloure-hi la seva adreça per tal de que el missatge de resposta segueixi el mateix camí. També s'inclou el paràmetre *branch* que serveix per identificar la transacció.

Més endavant es troba el **Max-Forwards**, aquest indica quin és el nombre màxim de salts que pot tenir el missatge fins arribar al destí. Aquest valor es va decrementant en cada salt.

To indica el nom d'usuari destinatari del missatge com també la seva SIP URI[8]

From molt similar al To, però aquest indica el remitent del missatge i a més conté un paràmetre anomenat *tag* (en altres missatges SIP la capçalera To també pot contenir aquest paràmetre). Aquest paràmetre és escollit per cada User Agent i serveix per identificar un diàleg entre dos usuaris.

Call-ID és un identificador únic per a tots el missatges enviats dins d'una mateixa sessió.

CSeq serveix per identificar i tenir un ordre dels missatges enviats. El valor s'incrementa en cada nou missatge enviat per l'usuari dins d'una mateixa sessió. Els missatges de resposta no incrementen aquest valor, sinó que mantenen el mateix valor que tenia el missatge que ha fet generar la resposta.

Contact indica la URI per contactar amb el remitent del missatge.

Content-type indica el tipus de contingut del cos del missatge.

[8] Uniform Resource Identifier, és una cadena de caracters que identifica unívocament un recurs.

Content-length indica la longitud del contingut del cos del missatge.

Un cop explicades breument les diferents capçaleres es descriuen, a continuació, alguns dels missatges SIP més comuns.

- **REGISTER** L'envia un usuari al un servidor SIP. Amb aquest missatge l'usuari s'identifica i indica quina és la seva localització en la xarxa.
- **INVITE** convida a un usuari a iniciar una sessió amb el remitent del missatge.
- **BYE** indica que l'usuari vol acabar la seva participació en la sessió.
- **MESSAGE** utilitzat per enviar missatges instantanis entre els usuaris d'una mateixa sessió.
- **200 OK** indica l'acceptació del missatge rebut per part d'un dels usuaris

2.1.2. Establiment d'una sessió SIP

El procés d'establiment d'una sessió SIP entre dos usuaris SIP es realitza mitjançant l'intercanvi de missatges.
A continuació es presenta el diàleg entre dos usuaris per establir una sessió multimèdia.

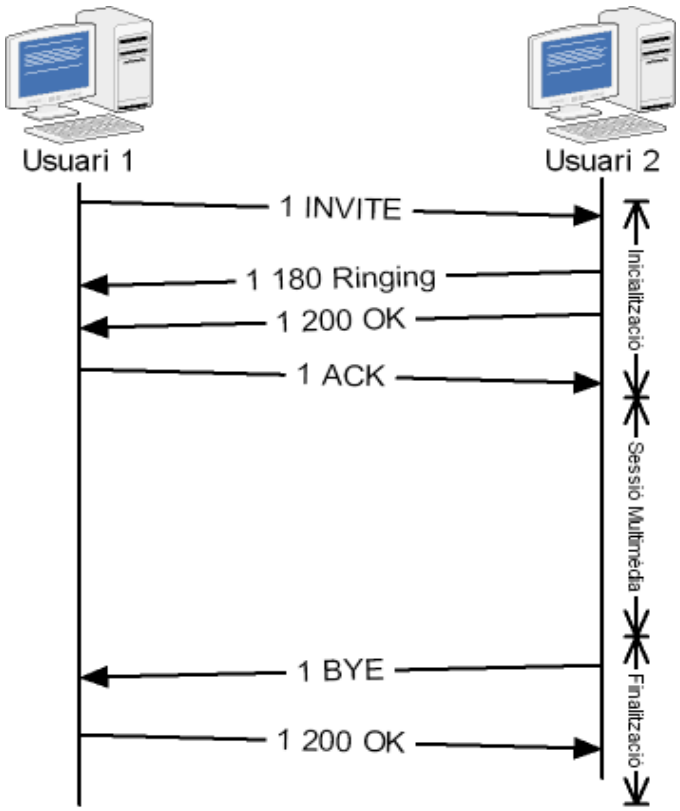

Fig. 2.2. Diàleg SIP bàsic

En la figura es poden observar alguns dels missatges SIP enviats entre els usuaris. El procés d'intercanvi d'aquests missatges és el següent:

L'usuari 1 vol iniciar una sessió amb l'usuari 2, per aquest motiu li envia un **INVITE**.

Mentre l'usuari 2 no accepta la invitació li envia a l'usuari 1 un missatge **180 Ringing** el qual indica que l'usuari encara no ha acceptat la trucada. Seria alguna cosa similar al to de trucada en un telèfon convencional.

Un cop l'usuari 2 accepta la invitació envia un **200 OK** per indicar-ho a l'usuari 1.

Llavors l'usuari 1 envia un **ACK** per confirmar que ha rebut l'acceptació.
Un cop arribats a aquest punt la sessió es considera establerta i a partir d'aquí es poden iniciar els serveis multimèdia que es desitgi.

Quan un dels usuaris vol abandonar la sessió, com li passa a l'usuari 2, envia un missatge **BYE**. Un cop el destinatari, usuari 1, rep el missatge li respon amb un **200 OK** tot indicant que ha acceptat la petició d'abandonament de la sessió.

2.1.3. Possibilitats de SIP

SIP inicialment es va definir amb l'objectiu de ser un protocol de senyalització i configuració de trucades en xarxes IP[9]. Però amb el temps s'ha vist que les aplicació d'aquest protocol pot ser extensible a totes aquelles aplicacions que facin ús de sessions multimèdia interactives com ara:

- Videoconferència
- Missatgeria Instantània
- Jocs online
- Localització de recursos

2.2. Introducció al Protocol RTSP

RTSP (Real Time Streaming Protocol) és un protocol, definit en l'RFC 2326[5], que permet l'establiment i el control d'un o més fluxos multimèdia. No s'encarrega de la entrega dels fluxos sinò del seu control, és, per tant, com un comandament a distància a través de la xarxa.

Algunes de les propietats que presenta RTSP entre altres són:

- RTSP té capacitat de multiservidor, fet que permet que cada flux multimèdia pertanyent en una sola presentació (en el cas d'una pel·lícula, el flux de vídeo i el d'àudio) puguin estar en màquines diferents.

[9] Internet Protocol, protocol no orientat a connexió destinat a la comunicació entre dos punts a través d'una xarxa de paquets conmutats (per exemple, Internet) .

- També té capacitat de negociació. Això permet indicar al client quines són les capacitats del servidor alhora de controlar el flux que ofereix.

2.2.1. Missatges RTSP

Per dur a terme el control del flux , RTSP fa ús d'una sèrie de missatges.
Tots aquests missatges, com en el cas de SIP, fan servir unes capçaleres molt similars a les de HTTP.

A continuació es presenta un exemple de missatge RTSP

```
SETUP rtsp://servidor.cat/documental.sd RTSP/1.0
Transport: RTP/AVP;unicast;client_port=4588-4589
```

Fig. 2.3. Missatge RTSP SETUP

En tots els missatges RTSP, com el de l'exemple anterior, sempre s'indicarà, a continuació del tipus de missatge, la URI del recurs multimèdia.

El missatge amb el qual el servidor dóna tota la informació sobre un recurs, és el missatge DESCRIBE. Un exemple de resposta a aquest missatge és el següent:

```
RTSP/1.0 200 OK
Content-Type: application/sdp
Content-Length: 376
i=Descripció del contingut¡
m=audio 3456 RTP/AVP 0
m=video 2232 RTP/AVP 31
```

Fig.2.4. Resposta a un missatge DESCRIBE

En aquesta resposta s'indica:

Content-Type, tipus de contingut del missatge, en el cas una descripció de sessió.

Content-Length, la mida del contingut.

I ara la part referent a la descripció:

- i , informació sobre el contingut

- m, descripció d'un flux multimèdia. Hi haurà tantes descripcions com fluxos formin el recurs. En la **Fig.2.4** hi apareix la descripció de dos fluxos, un de vídeo i l'altra d'àudio.

Alguns dels missatges RTSP més emprats són:

- **DESCRIBE**, com s'ha indicat anteriorment demana al servidor que enviï al client una descripció d'un recurs multimèdia.

Aquest missatge genera la resposta d'un 200 OK per part del servidor, on el contingut del qual és la descripció de cada un dels fluxos que componen el recurs.

- **SETUP**, vist en la **Fig.2.3.** , el client especifica com serà transportat el flux i quin serà el port per on el rebrà.
  Aquest missatge també genera un 200 OK per part del servidor. Una peculiaritat d'aquest 200 OK és que conté l'identificador de sessió, que serà únic per a cada sessió.

- **PLAY**, indica al servidor que li enviï el flux o fluxos del recurs multimèdia.

- **TEARDOWN**, el client indica al servidor que vol deixar de rebre els fluxos. Per tant, el servidor atura l'enviament de dades.

2.2.2. Establiment d'una sessió RTSP

Tal i com passa amb SIP, l'establiment d'una sessió RTSP es realitza a través d'un diàleg entre client i servidor, on s'hi estableix un intercanvi de missatges.
A continuació es presenta un exemple de diàleg entre un client i un servidor RTSP, on el s'estableix la sessió entre els dos punts, per a que finalment, el client rebi els fluxos multimèdia.

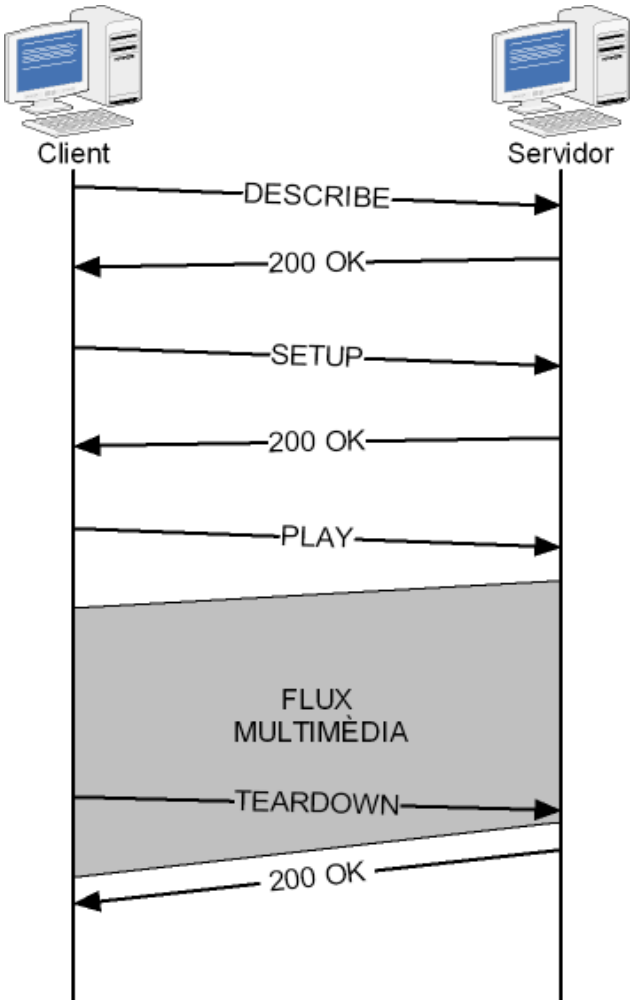

Fig.2.5. Diàleg RTSP

Es pot observar com el client envia al servidor un missatge **DESCRIBE** per tal de rebre en el **200OK** la descripció del recurs multimèdia.

Seguidament li envia un **SETUP** per establir el mode de transport i els ports recepció, com també per a que el servidor li assigni i li indiqui quin és l'identificador de sessió amb un **200OK**.

Un cop establida la sessió, el client ja pot demanar al servidor amb un missatge **PLAY** que li entregui els fluxos multimèdia.

Un cop el client vol deixar de rebre els fluxos li envia un **TEARDOWN** per tal de que el servidor deixi d'entregar-li els fluxos.

2.2.3. Aplicacions RTSP

Com s'ha anat comentant al llarg de l'apartat, RTSP és un protocol que permet el control de fluxos multimèdia. És per això que la principal aplicació d'RTSP és

en entorns on existeix una demanda de recursos multimèdia com seria VoD i el que es coneix com LiveStreaming.

- VoD (Video-on-Demand) o vídeo sota demanda és un sistema que permet a un usuari seleccionar i visualitzar un contingut de vídeo a través de la xarxa.
  Un exemple de VoD el trobem al conegut portal YouTube[10].

- LiveStreaming o streaming en viu, fa referència a tots aquells fluxos multimèdia que són introduïts a la xarxa quasi en el mateix instant que han estat generats per la font. És que utilitzen moltes emissores de ràdio i televisió que ofereixen la possibilitat de rebre la seva programació via Internet.
  Un exemple podria ser Catalunya Ràdio[11].

[10] YouTube Inc., www.youtube.com
[11] Catalunya Ràdio S.A., www.catradio.cat

CAPÍTOL 3. ARQUITECTURA DE XARXA

En aquest capítol es descriu quina és la arquitectura de xarxa present en l'ús dels protocols SIP i RTSP. L'anàlisi d'aquestes arquitectures, ajudarà a definir quina arquitectura és la adient per a la implementació del sistema especificat.

3.1. Arquitectura SIP

Els nombre d'elements que entren en joc en l'ús d'aquest protocol pot variar segons l'entorn en que es treballa. El nombre de dominis existents faran variar el nombre d'elements presents en l'escenari.

3.1.1. Un domini SIP

En el cas en que ens trobem en un entorn bàsic, és a dir, dins d'un únic domini SIP, el nombre d'elements amb els que ens podem trobar són els que es mostren en la següent figura:

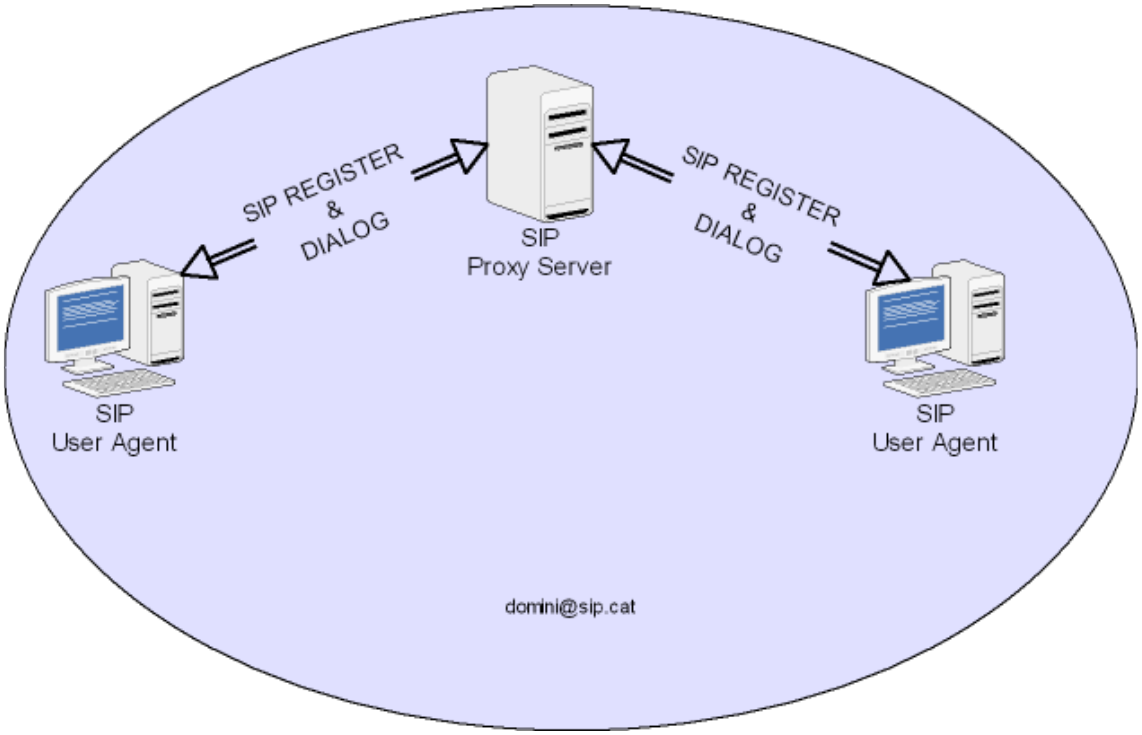

Fig. 3.1. Arquitectura en un domini SIP

Els diferents elements són:

- **SIP User Agents** (UAs) que són els dispositius extrems (PCs, telèfons, PDAs, etc) que creen i administren la sessió SIP.
  S'anomena User Agent Client qui envia els missatges i User Agent Server qui els rep i respon.

- **SIP Proxy Server** s'encarrega de rebre els missatge de registre de cada UA per tal de saber la localització de cadascun d'ells, com també s'encarrega de rebre els missatges dels UA Clients i entregar-los als UA Server i a l'invers.

3.1.2. Dos o més dominis SIP

En aquest escenari el nombre d'elements que entren en joc s'incrementa.
A continuació podem veure una representació gràfica d'un entorn amb dos dominis SIP:

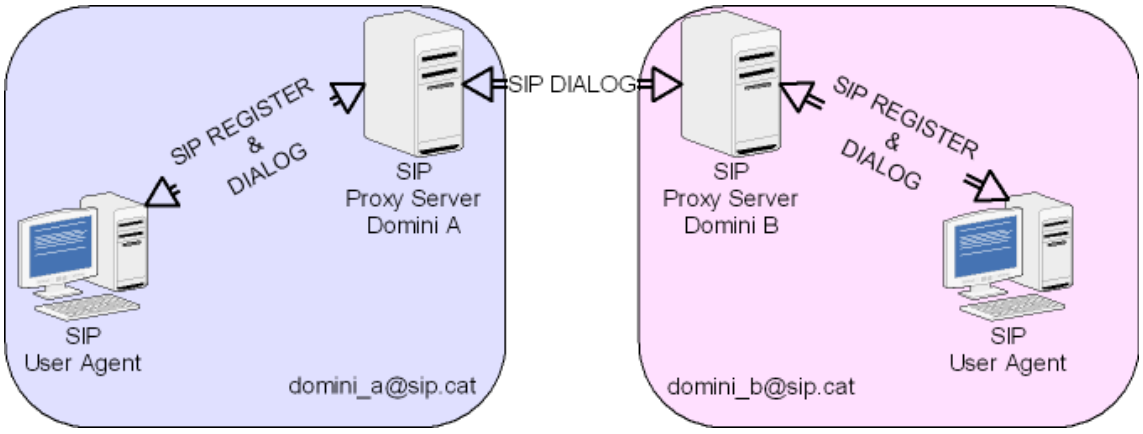

Fig. 3.2. Arquitectura en dos dominis SIP

En aquest escenari continuem trobant els dos elements bàsics en un entorn SIP, els SIP User Agents i els SIP Proxy Servers.
La diferència amb l'escenari anterior la trobem en que ara tenim dos SIP Proxy Servers. En general, tindrem tants SIP Proxy Servers com dominis SIP trobem.

Cada SIP Proxy Server de cada domini realitzarà les mateixes funcions que realitzaria en l'entorn d'un sol domini SIP. Però a més, s'encarregarà d'entregar els missatges, procedents del seu domini, amb destí a un segon domini, al SIP Proxy Server d'aquest domini.

3.2. Arquitectura RTSP

En un entorn RTSP els elements bàsics que trobem són:

- **RTSP Client**, qui envia els missatges de control de flux i qui rep el flux multimèdia que sol·licita del RTSP Server.

- **RTSP Server** és la màquina que rep els missatges de control dels RTSP Clients i els serveix el flux multimèdia que se li han sol·licitat.

A continuació presentem un exemple d'un flux multimèdia on el seu flux d'àudio i vídeo es troben en RTSP Servers diferents.

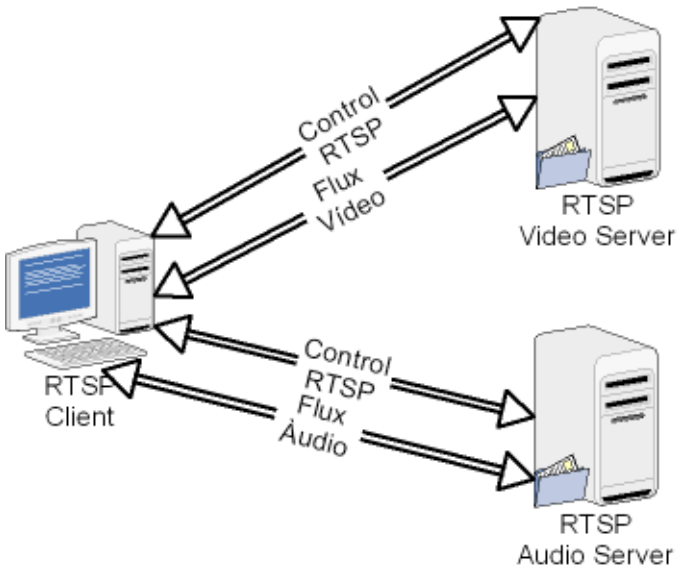

Fig. 3.3. Arquitectura RTSP amb fluxos allotjats en servidors diferents

El client és informat a través de la resposta d'un missatge DESCRIBE que el flux que vol rebre es troba dividit i que a més es pot trobar en un o més servidors. Gràcies a aquest informació el client és capaç de negociar per separat aquests fluxos i, per tant, també és capaç de rebre'ls de dos servidors diferents.

3.3. Arquitectura de xarxa del sistema

Prenent com a referència les arquitectures presents en les implementacions de SIP i RTSP, es pot tenir una primera idea de quina serà la arquitectura de xarxa de tot el sistema.
El resultat serà la unió de les dues arquitectures, com es pot apreciar en la següent figura:

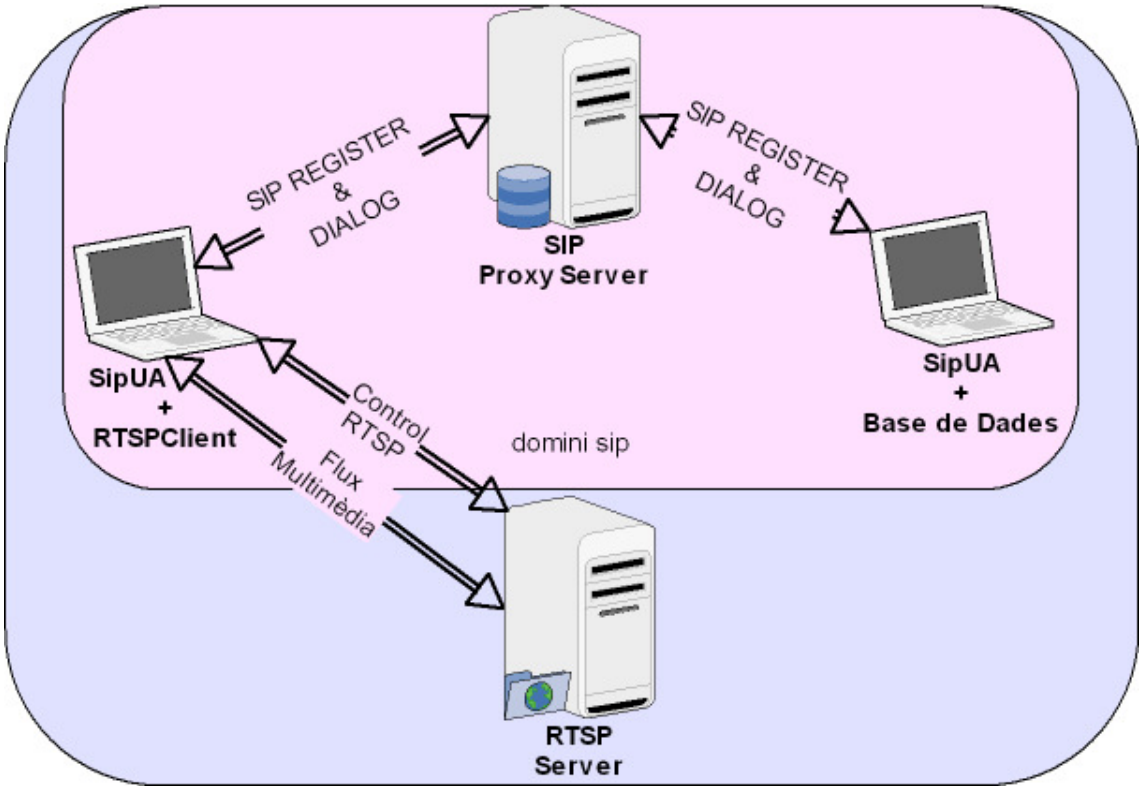

Fig. 3.4. Arquitectura de xarxa el sistema

Aquest esquema permet donar a conèixer quins seran els elements necessaris per al correcte funcionament del servei.
Els elements que formaran part del domini SIP són:

- **SIP Proxy Server**.
- El **client** que realitzarà funcions de SIP User Agent i client RTSP
- El **servidor** d'informació que realitzarà, també, funcions de SIP User Agent i que contindrà tota la informació referent als recursos multimèdia en una base de dades.
- El **servidor RTSP** encarregat de la distribució dels recursos multimèdia.

CAPÍTOL 4. DISSENY DEL SISTEMA

En el capítol anterior s'ha vist quina serà l'arquitectura de xarxa del sistema que es desenvoluparà i quins són els seus elements principals.
En aquest capítol es pretén justificar l'ús dels protocols SIP i RTSP, les funcionalitats de cada element i el disseny per a la implementació d'aquestes funcionalitats.

4.1. Solucions a problemes

Abans de començar a pensar en el disseny del servidor i del client del servei, s'ha de solucionar dos dels principals problemes que es presenten:

- La mobilitat del servidor.
- El control i entrega del flux multimèdia.

4.1.1. SIP com a protocol localitzador.

El principal problema que es pot trobar quan es parla de mobilitat dins la xarxa és conèixer quina és la localització de les màquines portàtils.
L'assignació dinàmica d'adreces IP i el possible canvi d'una xarxa a una altra, són els grans inconvenients alhora de fer localitzable una màquina.

L'estudi del protocol SIP ha permès descobrir que es tractava d'un protocol que permet de forma senzilla la localització de hosts dins d'un domini SIP.
És, a priori, un protocol que pot ser útil per ajudar a resoldre el problema de localització de màquines portàtils.

4.1.2. RTSP com a protocol de control de sessions multimèdia.

Un cop trobada la solució al problema de la mobilitat, cal pensar en com controlar l'entrega dels fluxos multimèdia.

Amb l'RTSP es va veure que conèixer el tipus de flux, iniciar la seva reproducció i aturar-la es reduïa a l'intercanvi de quatre missatges. Es va veure, doncs, que es tractava d'un protocol senzill i versàtil.

4.2. Elements del sistema

Un cop trobats els protocols que ajudaran en la localització de hosts i en l'establiment de sessions multimèdia, cal pensar en quins són els elements necessaris per a que tot el sistema funcioni.

En primera instància es sap que necessitem, fet que així ho requereix la implementació dels protocols, els següents elements:

- **SIP Proxy Server**. Com ja s'ha comentat en capítols anteriors, servirà per registrar totes les màquines clients i servidors participants.

- **RTSP Server** que contingui i serveixi els fluxos multimèdia que es vol distribuïr.

També serà necessari que els hosts que formin part del sistema i es vulguin beneficiar dels recursos multimèdia disposin d'un client, el qual s'anomenarà *mediaClient*, que permeti tant el diàleg SIP com l'RTSP. Caldrà, doncs, un client que contingui:

- **SIP User Agent**, per tal de ser capaç de comunicar-se amb el SIP Proxy Server i amb altres SIP User Agents.

- **RTSP Player** que permeti l'establiment de sessions multimèdia amb l'RTSP Server.

La funció del *mediaClient* serà la de permetre a l'usuari visualitzar la llista de recursos multimèdia disponibles i permetre el consum del recurs escollit pel client.

Finalment, i per acabar de formar el sistema, l'element mòbil: la màquina amb tota la informació dels recursos multimèdia presents en el sistema, la màquina que s'anomenarà *mediaServer*. Aquesta haurà de tenir:

- **SIP User Agent** que permeti tant el registre amb el SIP Proxy Server, com el diàleg amb els *mediaClient*.

- **Base de Dades** que contingui tota la informació referent als recursos multimèdia dels sistema.

La funció del *mediaServer* serà la de servir la informació que li exigeixi el *mediaClient* per satisfer les necessitats de l'usuari.

Seguint la arquitectura especificada en el capítol anterior, aquest seran, els elements presents en el sistema de distribució de recursos multimèdia:

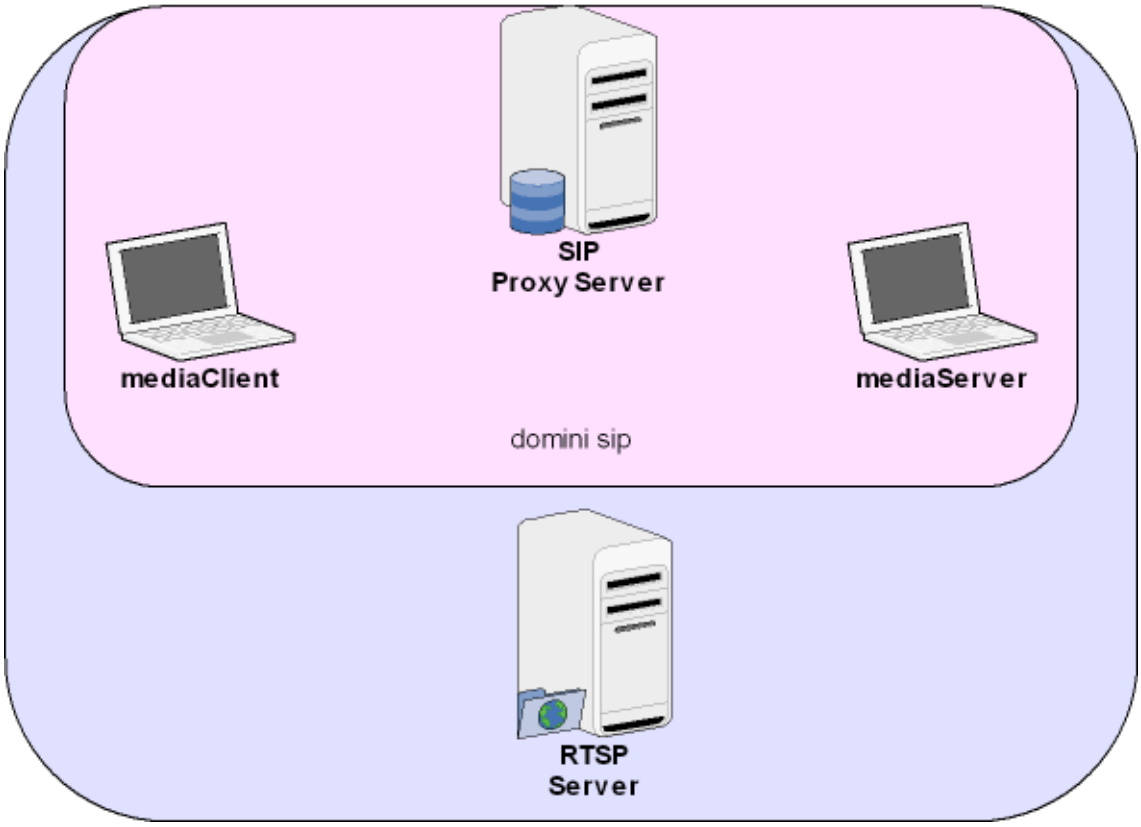

Fig.4.1. Elements del sistema.

4.3. Disseny esquemàtic dels elements del sistema

Un cop definits els elements que formaran part de la infraestructura de distribució, es defineixen, a continuació, les peces que els conformaran.

4.3.1. SIP Proxy Server

En aquesta màquina només ens caldrà tenir una aplicació que realitzi les funcions necessàries per a que funcioni com a SIP Proxy Server.

4.3.2. RTSP Server

A part d'una aplicació que realitzi les funcions de servidor d'RTSP es vol incloure un mòdul que permeti tenir no solsament un servei de Vídeo sota Demanda, el qual ja s'obtindria només amb un servidor RTSP, sinó també un servei d'streaming en viu o LiveStreaming, com per exemple un servei de ràdio.

Per poder gaudir d'un servei de LiveStreaming cal incloure un mòdul que permeti al servidor RTSP poder oferir aquest servei.

A continuació es presenta un diagrama de blocs del mòdul:

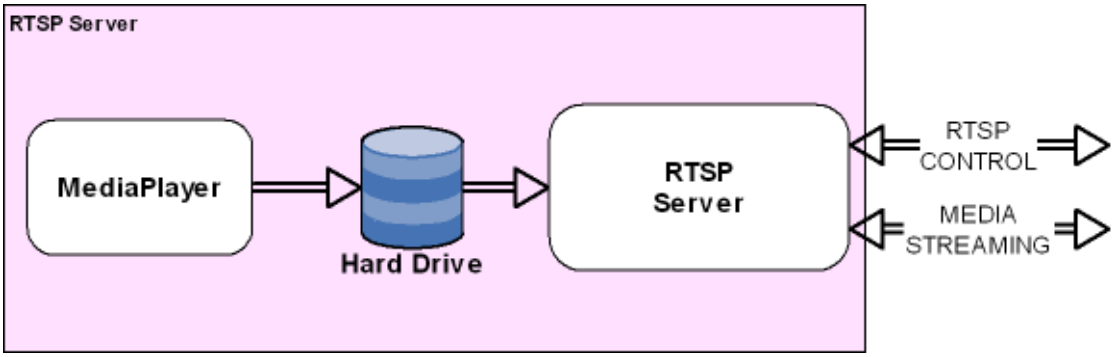

Fig.4.2. Diagrama de Blocs de l'RTSP Server

Es necessita d'un MediaPlayer o reproductor multimèdia en el que es tindrà la llista de reproducció de la programació que es vol oferir. S'escriurà a disc tota la informació de sortida del reproductor. A partir del fitxer escrit a disc dur, el servidor RTSP serà capaç d'oferir la informació procedent del reproductor.

4.3.3. Client multimèdia (MediaClient)

El diagrama de blocs del client del sistema és el següent:

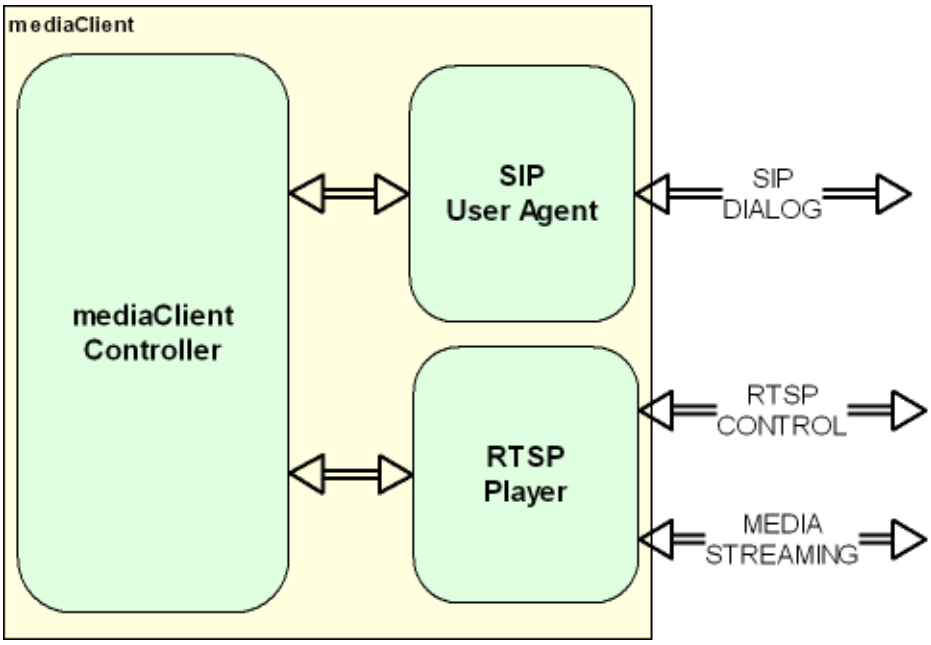

Fig.4.3. Diagrama de Blocs del MediaClient

Com ja s'ha comentat, el mediaClient contindrà un SIP User Agent, un RTSP Player i, a més, un mòdul que controli tota la lògica de funcionament del client anomenat **mediaClient Controller**.

El **mediaClient Controller** s'encarregarà de:

- Indicar al SIP User Agent quan i com s'ha de registrar al SIP Proxy Server.
- Comunicar-se, a través del SIP User Agent, amb el mediaServer. És a ell qui li demanarà tota la informació necessària per a que pugui rebre un flux multimèdia.
- Indicar al RTSP Player la URI del recurs multimèdia que es vol obtenir per tal de que aquest es posi en contacte amb el RTSP Server i puguin començar la negociació d'entrega del flux.

A través del **SIP User Agent** es podran enviar i rebre missatges i comandes pròpies del sistema a través de missatges SIP estàndards[12].

L'**RTSP Player** permetrà a l'usuari visualitzar els recursos multimèdia que sol·liciti.

4.3.4. Servidor multimèdia (MediaServer)

A continuació es presenta el diagrama de blocs del mediaServer:

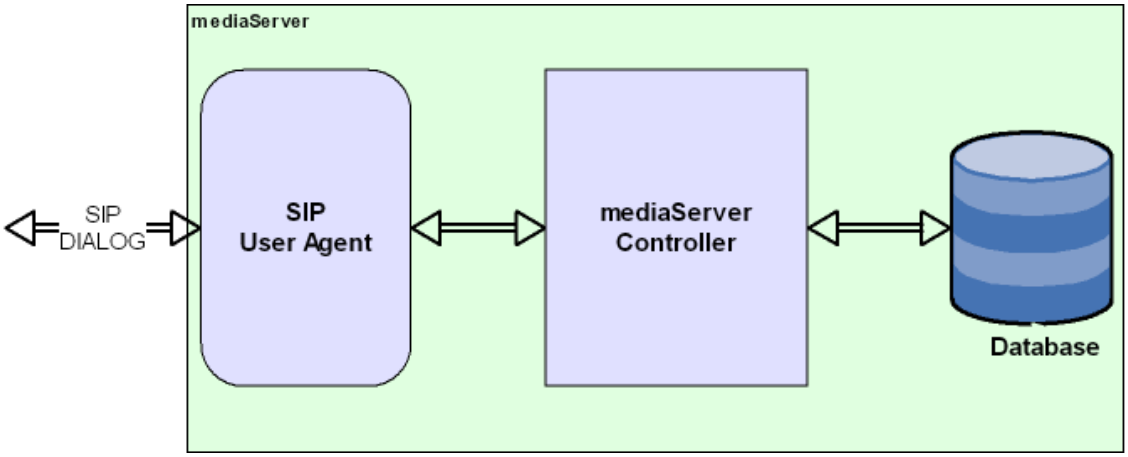

Fig.4.4. Diagrama de Blocs del MediaServer

Tal i com passa amb el mediaClient, el mediaServer també necessita d'un mòdul que permeti controlar tota la lògica de funcionament, en aquest cas el **mediaServer Controller**.

El **mediaServer Controller** s'encarregarà de:

- Indicar al SIP User Agent quan i com registrar-se al SIP Proxy Server.

[12] Tipus de missatges propietari, veure Annex B.

- Gestionar els missatges rebuts pel SIP User Agent i actuar en conseqüència.

- Realitzar les consultes a la base de dades per tal de satisfer la demanda dels clients.

CAPÍTOL 5. IMPLEMENTACIÓ DEL SISTEMA

En el capítol anterior s'ha definit quins són i què faran els elements que integraran el sistema. Un cop aclarides les funcions de cada element s'exposa, a continuació, quina ha estat la seva implementació.

5.1. Escenari d'implementació

La implementació del sistema s'ha realitzat tenint en compte l'escenari presentat en la definició del disseny.
L'escenari amb el programari necessari per a la implementació de cada element és el següent:

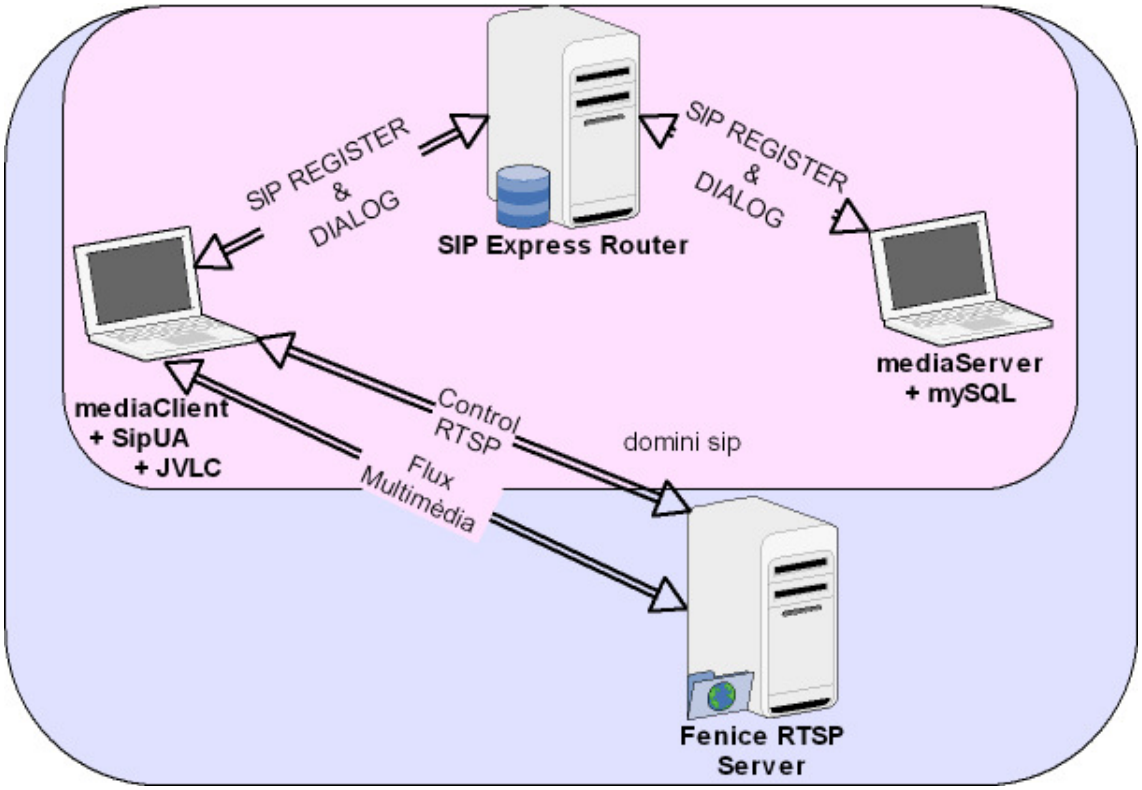

Fig.5.1. Representació de l'escenari implementat amb el programari emprat.

En els següents apartats es descriu pas a pas la implementació de cada element present en el sistema.

5.2. Implementació del SIP Proxy Server

La funció de SIP Proxy Server la pot realitzar qualsevol màquina que suporti el programari adient per realitzar aquesta tasca.

5.2.1. El programari

L'entorn en el que es va desenvolupar aquest treball, dins la fundació i2Cat, ja tenia en funcionament una màquina amb les característiques de SIP Proxy Server, i per tant, no calia desenvolupar cap software en aquest sentit.
El software utilitzat és el SIP Express Router de iptel.org, més conegut com a SER[7].

SIP Express Router és un programari sota llicència GNU i que pot ser executada sota qualsevol distribució del sistema operatiu Linux.

Les principals característiques que suporta, entre altres, són:

- Localització d'usuaris SIP
- Missatgeria
- Temporitzador de sessió
- Tenir accés a sistemes de base de dades com mysql, oracle, radius, entre altres.

De totes les característiques que ofereix SER, les que són de més utilitzat són:

- La localització d'usuaris SIP, per poder localitzar el MediaServer
- La possibilitat d'enviar missatges. Així facilita l'enviament de comandes entre participants.

5.2.2. Posada en marxa

La configuració i posada en funcionament és senzilla, només cal executar la comanda adient un en terminal Linux i la màquina ja estarà preparada per actuar com a SIP Proxy Server.

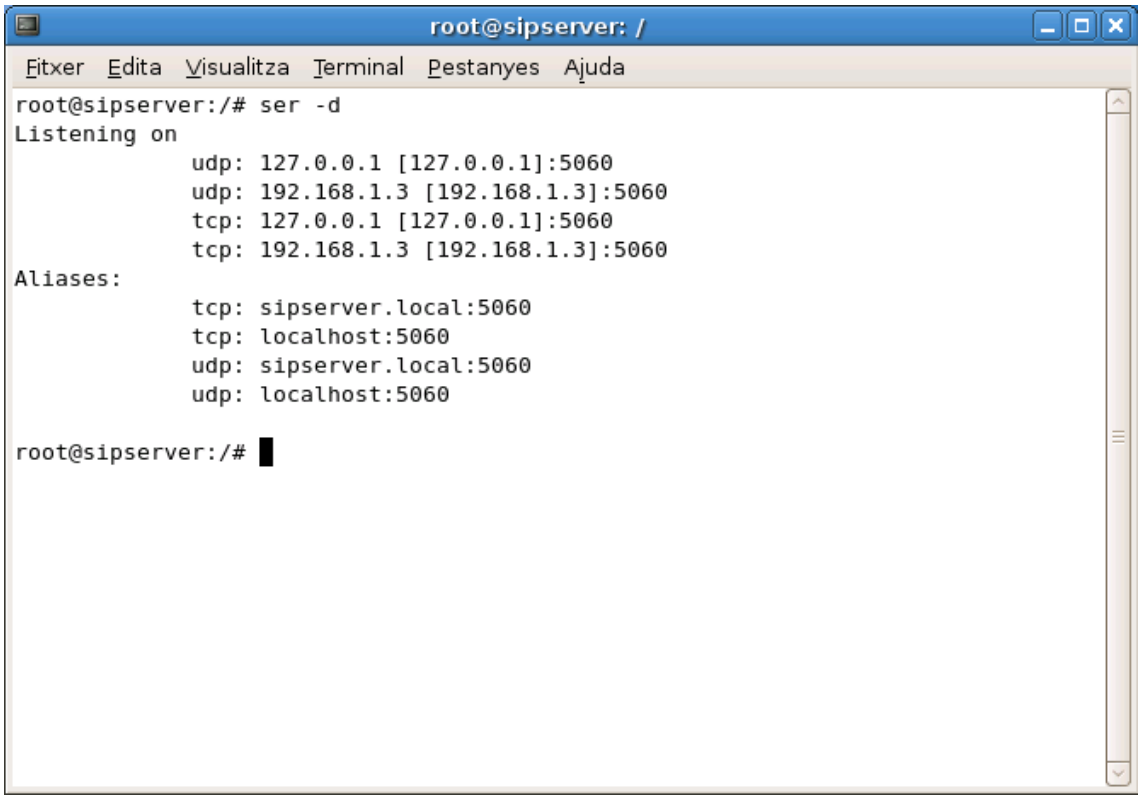

Fig.5.2. Captura de pantalla. Arrencada del SER

En la figura es pot observar com s'informa de que la màquina està pendent de possibles comunicacions SIP. També informa de quins són els ports pels quals està esperant comunicació i el tipus de protocol de transport de cada port.

SIP Express Router permet visualitzar en tot moment el llistat d'usuaris SIP registrats. Això és útil per saber en tot moment qui es troba registrat en el domini SIP al qual pertany. Així es pot portar un control dels usuaris presents en la xarxa. Es pot veure un exemple d'aquest llistat en la següent figura:

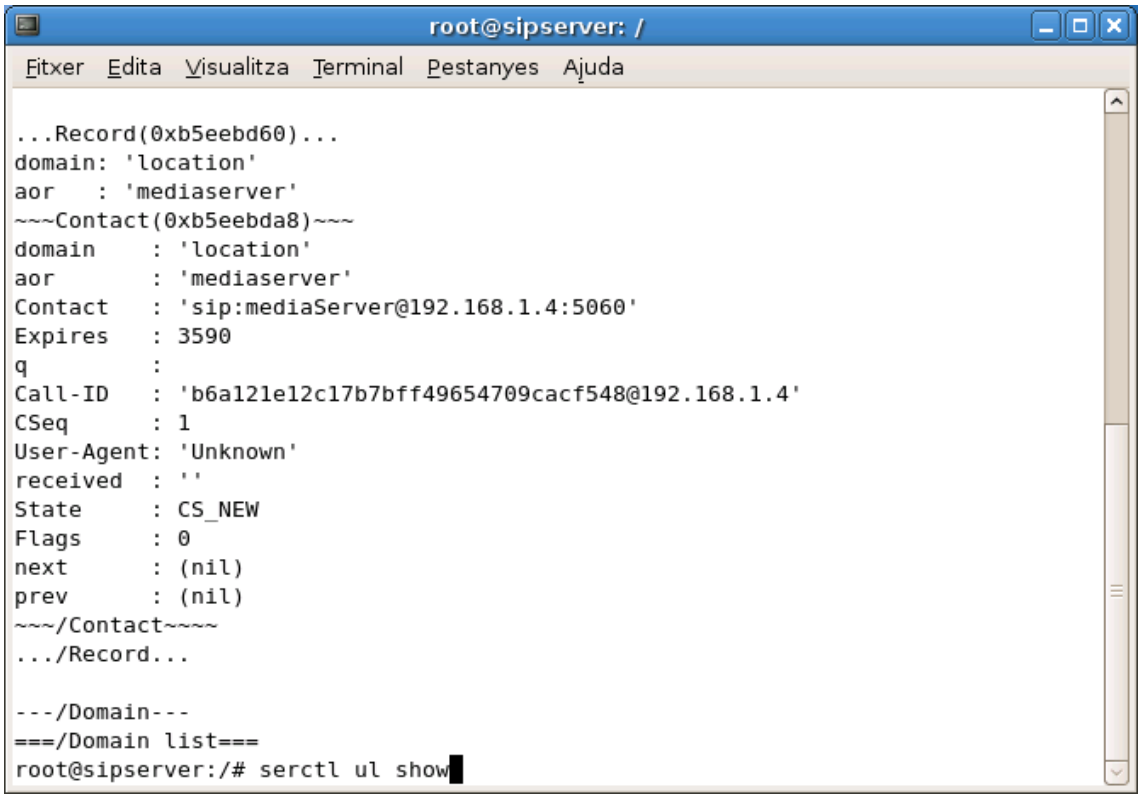

Fig.5.3. Llistat d'usuaris SIP registrats en el SER

A partir d'aquest moment, ja està disponible el SIP Proxy Server. Per tant, ja està disponible el domini SIP en el que es treballarà durant tot el procés.

5.3. Video sota Demanda. Implementació del RTSP Server

Per donar el servei de VoD o vídeo sota demanda cal implementar un servidor RTSP. Com en el cas anterior, només cal d'un programari adient per tal de que una màquina funcioni com a servidor RTSP. Però, a diferència amb el cas anterior, cal més que un software per a que el RTSP Server del sistema pugui servir fluxos LiveStreaming.
Com ja s'ha vist en el disseny d'aquest element, l'RTSP Server es dividirà en dos mòduls. El primer realitzarà les funcions pròpies d'un servidor RTSP, donar servei de VoD i el segon prepararà l'RTSP Server per a que pugui servir fluxos LiveStreaming.

5.3.1. Servidor RTSP

Per a que el RTSP Server realitzi funcions de servidor RTSP bàsic, només cal la configuració i instal·lació del programari adient.

Després d'estudiar les possibilitats ofertes per empreses i organitzacions, es va decidir per un software desenvolupat pel Politècnic de Torí[13], el **Fenice**[11].

Fenice és un servidor de fluxos multimèdia que implementa el protocol RTSP. Suporta els següents estàndards de codificació[14]:

- MP3 (MPEG-1 Layer III) (RFC 3119) per a àudio
- MPEG-1/2 i MPEG-4 per a vídeo.

Les seves principals característiques són:

- Adaptabilitat a l'estat de la xarxa
- Capacitat per administrar sessions de LiveStreaming amb l'ajut d'eines externes.
- Suport de llicències Creative Commons[15] per a l'streaming d'àudio i vídeo.

Els dos punts forts que han fet decantar la balança cap a la tria d'aquest software són: la compatibilitat amb els estàndards de codificació i la capacitat de gestionar sessions de LiveStreaming.

5.3.2. Fitxers SD

Una de les característiques principals en una sessió RTSP són les descripcions del contingut d'un flux multimèdia. Per a que el servidor sigui capaç de compartir aquesta informació és necessari que aquest posseeixi el fitxer SD referent al flux a servir.

Els fitxers SD contenen tota la informació necessària per a que el servidor RTSP pugui entregar el flux i per a que el client el pugui reproduir. Aquests fitxers estan construïts a base de tags o etiquetes. I cada etiqueta fa referència a una característica del recurs multimèdia.

A l'Annex C es pot trobar tota la informació referent a aquest tipus de fitxer.

5.3.3. Configuració i posada en marxa del RTSP Server

Fenice és Open Source i el seu codi font és pot descarregar des de la seva pàgina web.

Prèviament a la seva execució cal compilar el codi font i instal·lar-ho[16] sobre qualsevol distribució Linux. La distribució Linux utilitzada és Ubuntu 7.04[17].

Un cop realitzada la compilació i instal·lació del Fenice cal configurar-lo segons les necessitats que es tinguin. Cal indicar:

- Directori on s'emmagatzemen els fitxers SD.

[13] Politecnico di Torino: http://www.polito.it/
[14] Veure Annex C
[15] Creative Commons és una **organització que ofereix un sistema de drets d'autor.**
[16] Com instal·lar Fenice, veure Annex A

- Port RTSP per on rebre les connexions.
- Nombre màxim de peticions que podrà atendre la màquina.

Un cop realitzades totes les tasques de configuració, el Fenice es troba a punt per poder ser iniciat[17] i per tant poder iniciar el servei RTSP dins del sistema.

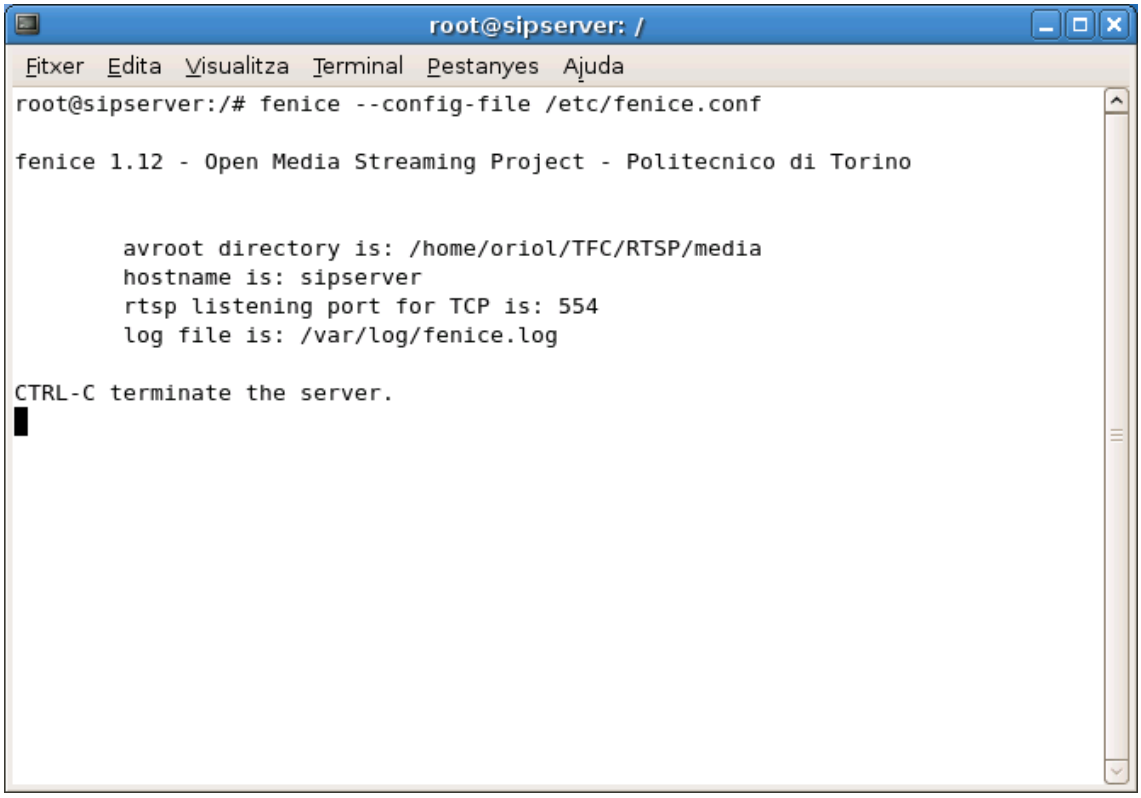

Fig.5.4. Inici del servidor Fenice

5.3.4. Proves prèvies

Abans d'integrar el servidor RTSP en el sistema de distribució cal realitzar proves per comprovar el correcte funcionament.
Les proves són senzilles i només requereixen d'un client RTSP en una màquina remota, o la mateixa local.

El client RTSP utilitzat és el VideoLan[12].
A continuació es pot veure una captura de pantalla en el moment de realitzar les proves amb el VideoLan:

[17] Com iniciar Fenice, veure Annex A

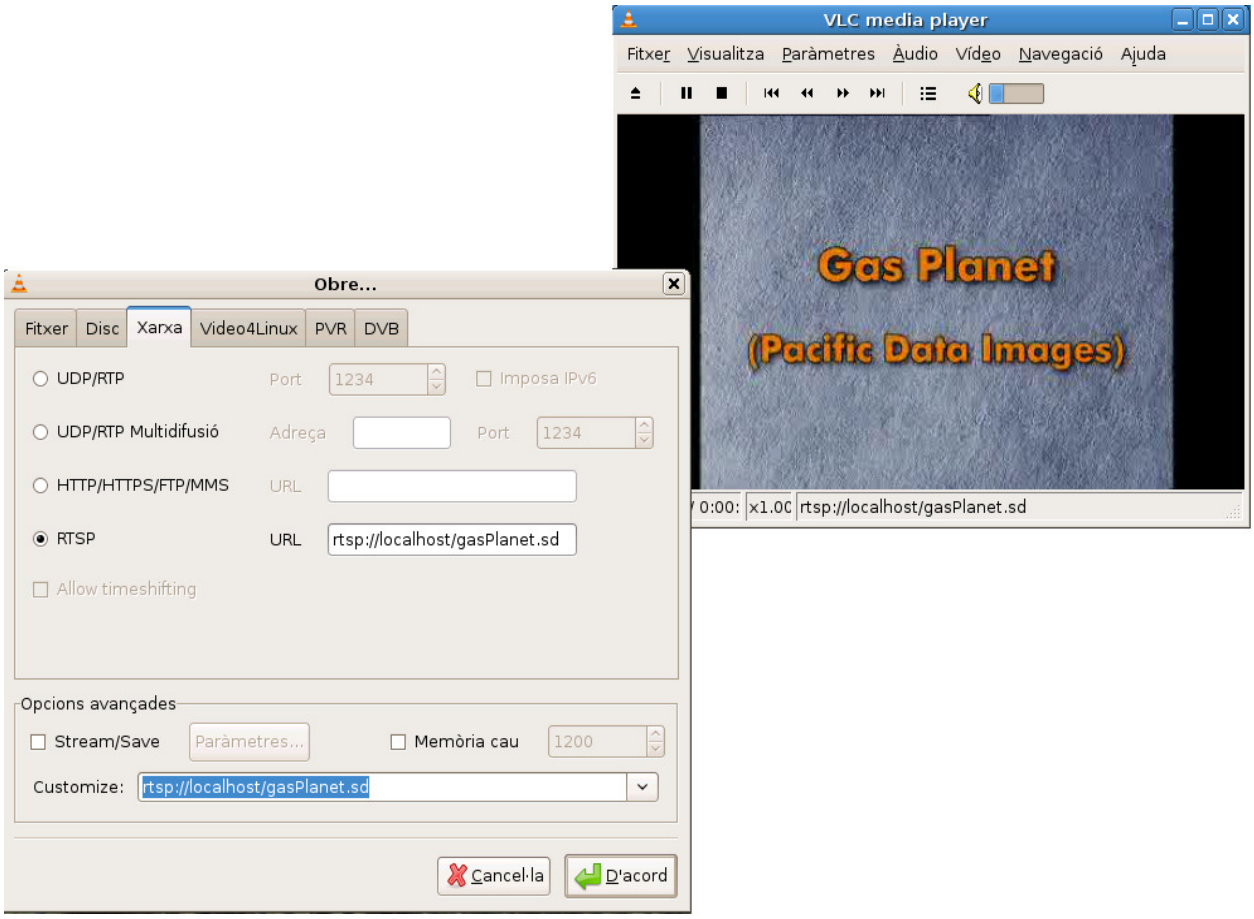

Fig.5.5. Proves amb el client RSTP VideoLan.

5.4. Incorporació del servei de LiveStreaming

Per a que Fenice pugui oferir el servei de LiveStreaming cal realitzar una sèrie de passos de configuració.

Primerament caldrà instal·lar el codificador **ffmpeg**[18]. Aquest codificador permet codificar en un arxiu de sortida qualsevol flux d'àudio i/o vídeo procedent de qualsevol dispositiu de la màquina, com per exemple una webcam o sintetitzador.

Un cop instal·lat el codificador s'ha de crear un fitxer especial FIFO[18]. Aquests fitxers funcionen com a interconnexions o canonades (més conegudes pel seu nom en anglès, *pipes*) entre dos processos. Això permet a un procés realitzar funcions d'escriptura i a un altre procés funcions de lectura.

Amb aquests dos elements s'aconsegueix que tot el flux procedent de dispositius d'àudio o vídeo siguin codificats en un fitxer que al mateix temps pugui ser llegit pel servidor Fenice.

[18] Per crear un fitxer FIFO, veure Annex A

A partir d'aquest moment, tots els fluxos de sortida dels dispositius d'àudio i vídeo de la màquina poden ser servits pel servidor RTSP.
Ara només caldrà triar els continguts. Per exemple, amb l'ajuda d'un reproductor com el VideoLan es pot reproduir una llita de reproducció amb tota una selecció de música.

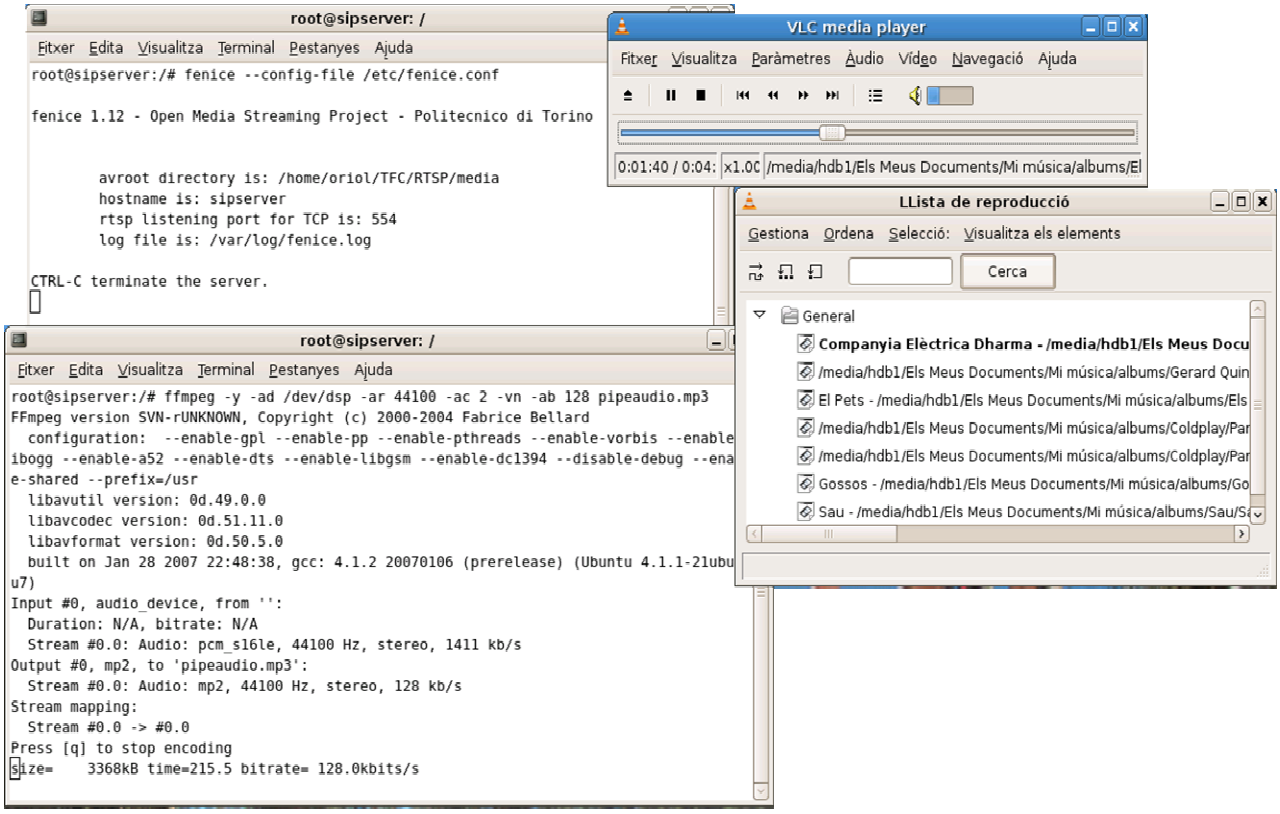

Fig.5.6. Terminal d'execució del servidor Fenice amb el servei de LiveStreaming i del reproductor VideoLan.

5.5. Implementació del servidor multimèdia o MediaServer

Com ja s'ha comentat en capítols anteriors, el MediaServer és l'element que informa de la localització dels recursos multimèdia del domini. També s'ha vist que aquest element té la característica de poder ser portable i estar localitzat en qualsevol lloc de la xarxa.

A continuació s'exposa quina ha estat la implementació de cada un dels blocs que conformen el MediaServer, a excepció del SIP User Agent, que s'explicarà en un apartat específic, per tractar-se d'un element comú del MediaServer i MediaClient.

5.5.1. Mòdul de gestió de dades.

Per poder informar sobre els continguts presents en el domini cal ser propietari d'aquesta informació. Una de les vies per tenir a l'abast tota la informació que es desitgi, és amb la implementació d'una base de dades.

En el MediaServer s'ha implementat la base de dades mitjançant el sistema de gestió de base de dades MySQL[21].

El model entitat/relació utilitzat en la implementació de la base de dades és el següent:

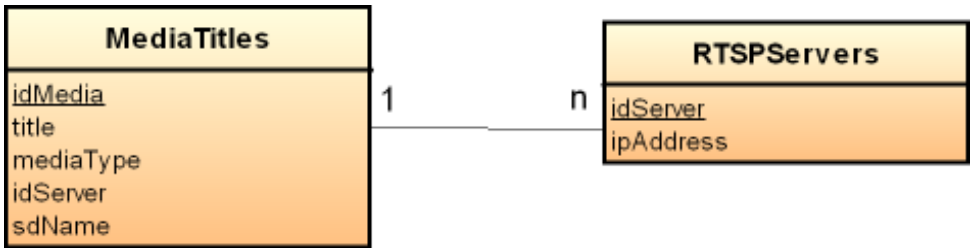

Fig.5.7. Model E/R de la base de dades

Per tant les taules de la base de dades quedaran de la següent forma:

mediatitles:

idMedia P.K.	title	mediaType	idServer F.K.	sdName

rtspservers:

idServer P.K.	ipAddress

Fig.5.8. Taules de la base de dades

El significat de cada atribut és el següent:

- En la taula *mediatitles*:
  - **idMedia** és l'identificador únic (Parent Key) del recurs multimèdia.
  - **title** indica el nom del recurs, per exemple, el títol d'una pel·lícula en el cas que ho sigui.
  - **mediaType** indica de quin tipus de recurs multimèdia es tracta: àudio, vídeo o ràdio (LiveStreaming).
  - **sdName** és el nom de l'arxiu SD del servidor RTSP que conté tota la informació referent al recurs.
  - **idServer** fa referència a l'identificador del servidor o servidors RTSP que contenen aquest recurs. Aquest atribut està vinculat a l'atribut idServer de la taula *rtspserver* (Foreign Key).

- En la taula *rtspservers*:
    - **idServer** és l'identificador únic (Parent Key) del servidor RTSP.
    - **ipAddress** indica l'adreça IP del servidor.

Amb tots aquests paràmetres el MediaServer té la informació necessària per informar al MediaClient dels recursos disponibles i quines són les seves localitzacions.

Per tenir accés a la base de dades des de qualsevol aplicació implementada en java cal fer ús de la llibreria mysql-connector. Aquesta llibreria conté tots els mètodes necessaris per realitzar lectures i escriptures en la base de dades.

En el MediaServer s'han implementat, amb l'ajut de la llibreria mysql-connector, dues classes que permetran al MediaServerController a realitzar les consultes a la base de dades. Aquestes classes són:

- **DataBaseConnection**: permet establir la connexió amb la base de dades per tal de poder realitzar les accions oportunes.

- **DataBase**: té en compte la estructura de la base de dades del MediaServer i permet obtenir la llista de tots els recursos multimèdia segons del tipus que siguin, com també permet obtenir informació de localització d'un recurs en concret.

5.5.2. Mòdul controlador (MediaServer Controller)

Com en tot programari, és essencial la incorporació d'un element que porti tota la lògica de funcionament i que, a més, sigui capaç d'interactuar amb els demés elements.

En el MediaServer, aquesta funció la realitza el *MediaServer Controller*. Com ja s'ha comentat en el capítol anterior, aquest element servirà d'interconnector entre el SIP User Agent i la base de dades.

Aquest element s'ha implementat en una classe java anomenada *MediaServerController*. El seu diagrama de classes és el següent:

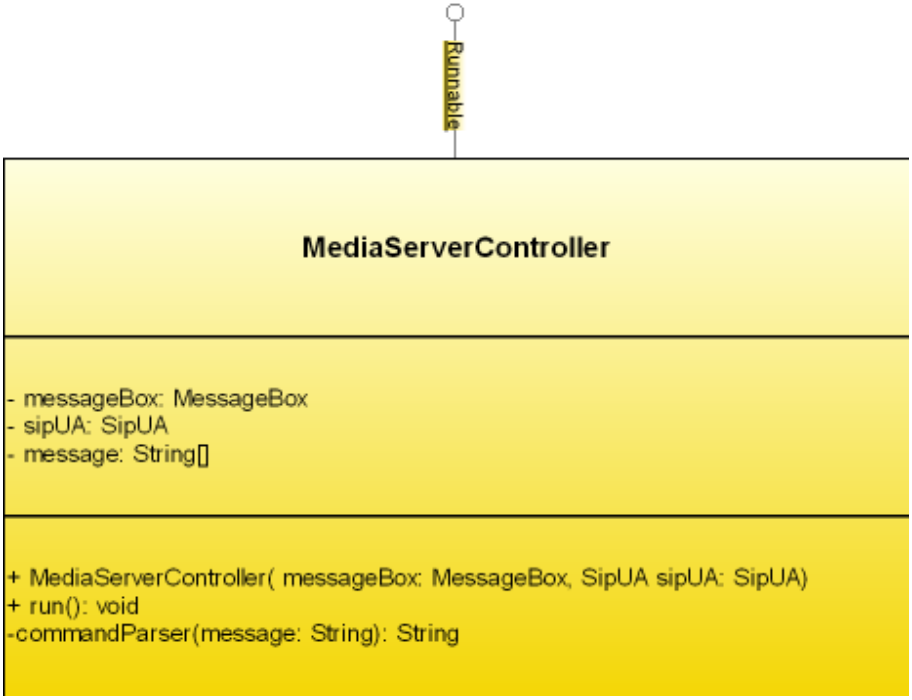

Fig.5.9. Diagrama de classe del MediaServerController

Dos dels atributs utilitzats, sipUA i messageBox són referències a altres dos atributs del mateix tipus que es troben en el MediaServer. L'altre, message, és on es desarà el missatge rebut pel SIP User Agent sipUA.

La lògica de funcionament és la següent:

- El MediaServer Controller espera fins que el SIP User Agent rep un missatge i el desa en el MessageBox.

- Un cop rebuda l'alerta de que hi ha un nou missatge en el MessageBox, el MediaServer Controller extreu el nou missatge del messageBox i analitza el contingut.

- Segons el contingut del missatge[19] realitzarà la consulta adient a la base de dades.

- Un cop realitzada la consulta donarà l'ordre al SIP User Agent (sipUA) d'enviar un missatge on el seu contingut sigui el resultat de la consulta.

5.5.3. Servidor Multimèdia o MediaServer

El servidor multimèdia o MediaServer està format pel mòdul de gestió de dades, el mòdul controlador i el Sip User Agent. La unió de tots els elements del MediaServer s'ha aconseguit a través de la implementació d'una classe anomenada de la mateixa forma, MediaServer. Aquesta classe modela el

[19] Tipus de missatges, veure Annex B

MediaServer amb totes les funcions descrites anteriorment. El seu diagrama de classe és el següent:

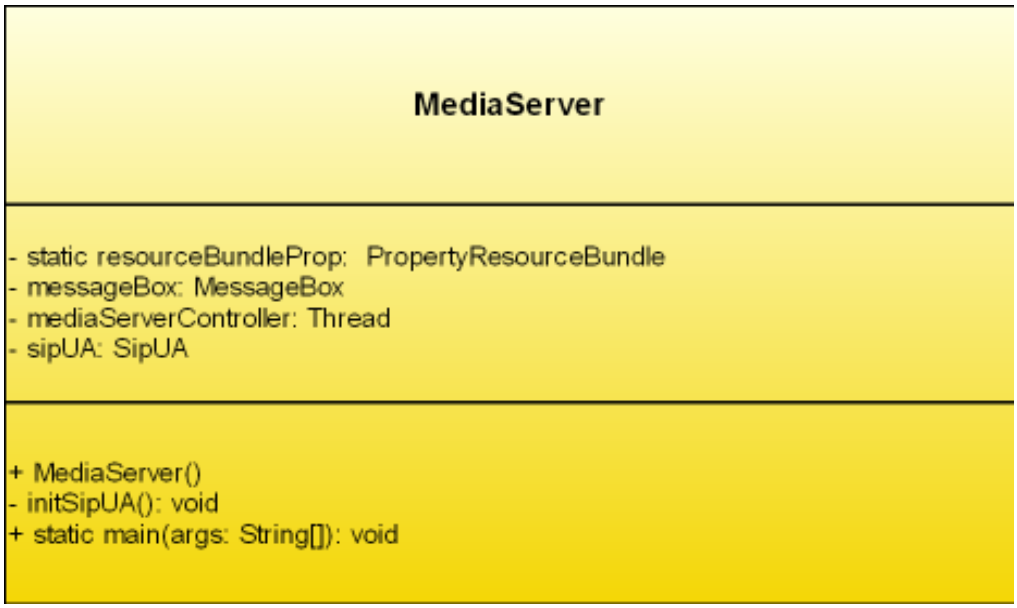

Fig.5.10. Diagrama de classe del MediaServer

Aquesta és la descripció de cada un dels atributs:

- **resourceBundleProp** permet la configuració del *MediaServer* a través d'un fitxer de configuració. Es poden configurar tots els elements referents al SIP User Agent:
  - El nom d'usuari dins del domini SIP.
  - L'adreça IP del SIP Proxy Server
  - El port utilitzat pel SIP User Agent per comunicar-se amb altres usuaris.
  - El tipus de protocol de transport: udp o tcp.

- **messageBox** és el contenidor de missatges entrant. És la "safata" on el *sipUA* desarà els missatges rebuts per tal de que el *MediaServer Controller* els pugui llegir.

- **mediaServerController**, comentat en l'apartat anterior, és el responsable del bon funcionament del *MediaServer*.

- **sipUA** és el SIP User Agent del *MediaServer*. En apartats posteriors es descriu la seva implementació.

Tot el conjunt fa que el sistema ja disposi d'un dels seus elements, el *MediaServer*.

5.6. Implementació del client multimèdia o MediaClient

El MediaClient, com s'ha comentat anteriorment, és l'element beneficiari del sistema. És el client receptor dels recursos multimèdia disponibles.
Per dur a terme la seva implementació s'ha tingut en compte el disseny inicial presentat en l'apartat 4.3.3 d'aquest mateix document.

A continuació es desenvolupa detalladament la implementació del MediaClient com també la implementació de tots el mòduls que el conformen.

5.6.1. Mòdul controlador (MediaClient Controler)

De la mateixa manera que en el MediaServer es necessitava d'un mòdul capaç de dur tota la lògica de funcionament i capaç, també, d'interconnectar els demés mòduls, en el MediaClient es troba la mateixa necessitat. Aquesta necessitat es cobreix amb el MediaClient Controller.

El MediaClient Controller s'ha implementat mitjançant una classe anomenada d'igual manera, MediaClientController.

El diagrama de classe del MediaClientController és el següent:

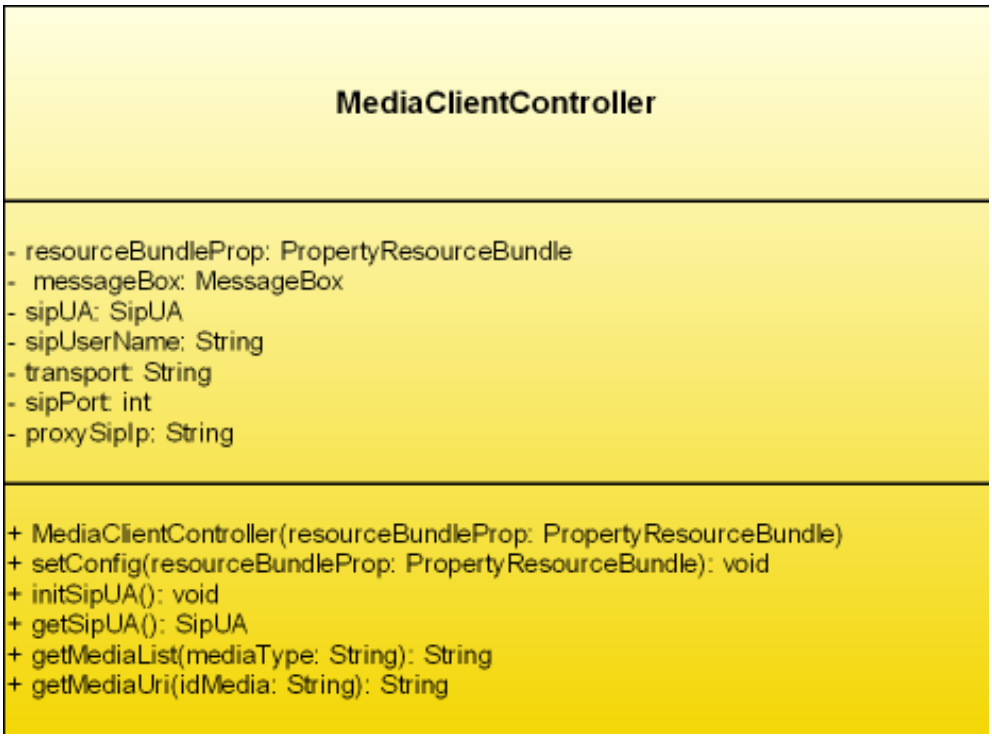

Fig.5.11. Diagrama de Classe del MediaClientController

Els atributs són els mateixos que trobem en el seu homòleg, el MediaServerController:

- **resourceBundleProp** per a la configuració mitjançant fitxers de configuració.
- **messageBox** per llegir els missatges rebuts pel SIP User Agent
- **sipUA** com a SIP User Agent.

Les dues funcions que realitza el MediaClientController són:

- Obtenir la llista del recursos multimèdia del servidor segons els tipus.
- Obtenir del servidor, la localització d'un recurs.

Totes aquestes funcions les realitzarà amb l'ajut del sipUA, gràcies al qual es pot establir comunicació amb el servidor.

5.6.2. La interfície gràfica

La gran diferència que hi ha entre el MediaServer i el MediaClient és que el funcionament del primer és automàtic, mentre que el segon actuarà segons la voluntat de l'usuari que en fa ús.

Per tal de que un usuari sigui capaç d'utilitzar i configurar el MediaClient, s'ha desenvolupat una interfície gràfica.
A través d'aquesta interfície l'usuari serà capaç de donar les ordres adients al MediaClient Controller per tal de poder gaudir de la recepció dels recursos multimèdia.

Mitjançant la interfície gràfica es poden configurar paràmetres referents al SIP User Agent:

- Port a través del qual el MediaClient es comunica amb el MediaServer i SIP Proxy Server
- Adreça IP del SIP Proxy Server.
- Protocol de transport utilitzat en les comunicacions SIP.

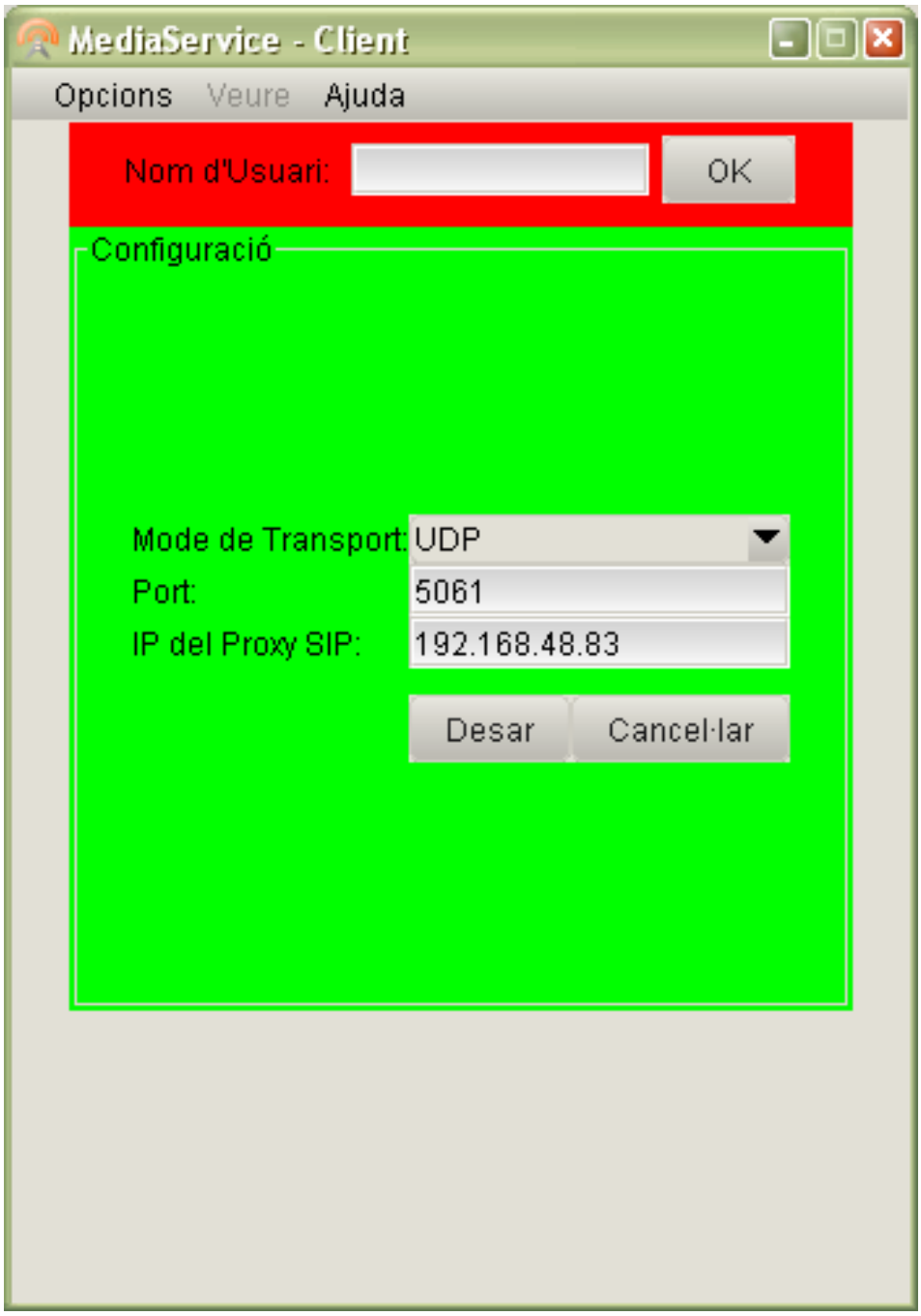

Fig.5.12. Finestra de configuració del MediaClient

L'usuari també serà capaç d'escollir un recurs multimèdia de la llista de recursos disponibles.

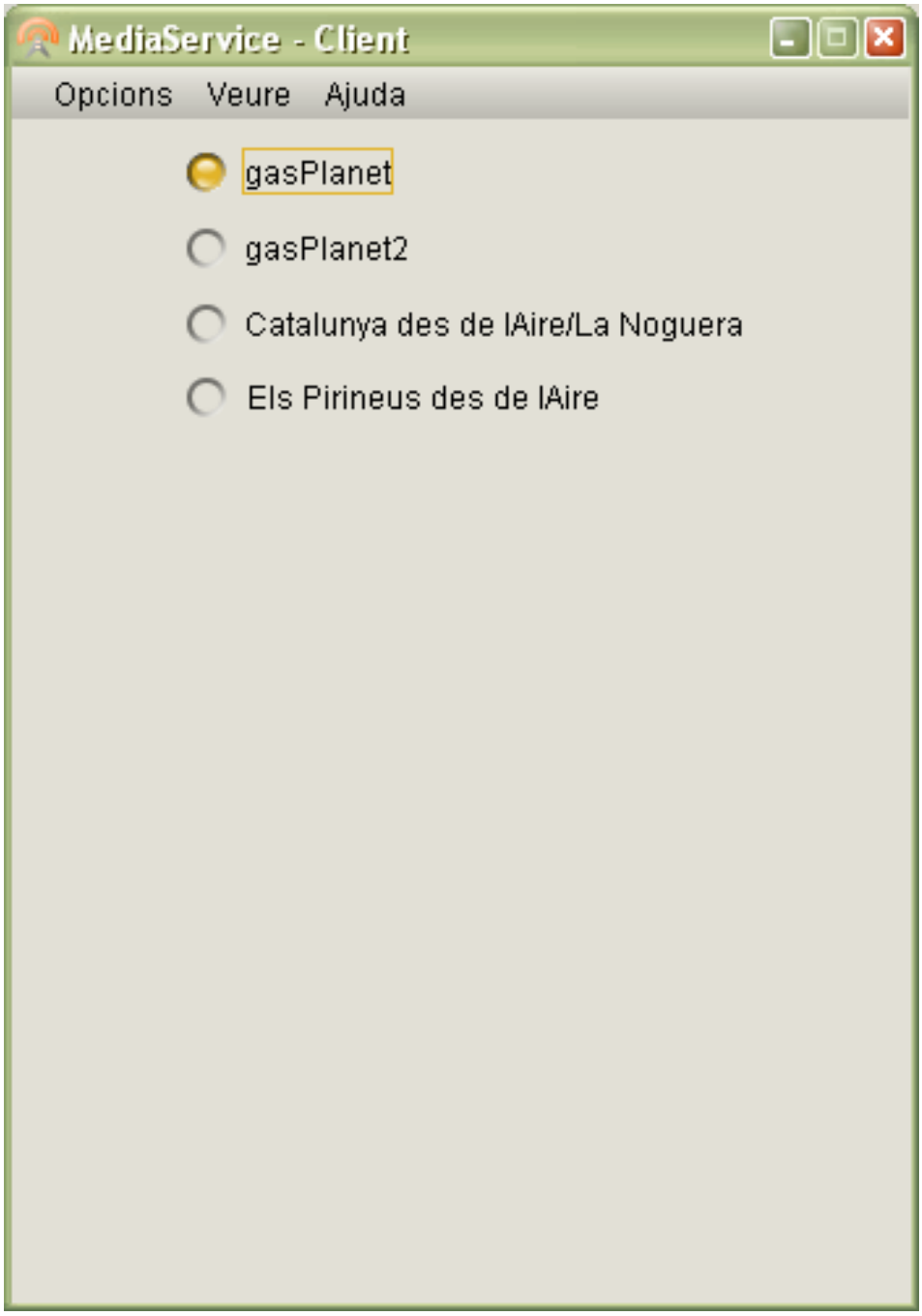

Fig.5.13. Llista de vídeos disponibles.

Gràcies al reproductor de vídeo integrat, l'usuari podrà rebre el recurs i visualitzar-ne el contingut.

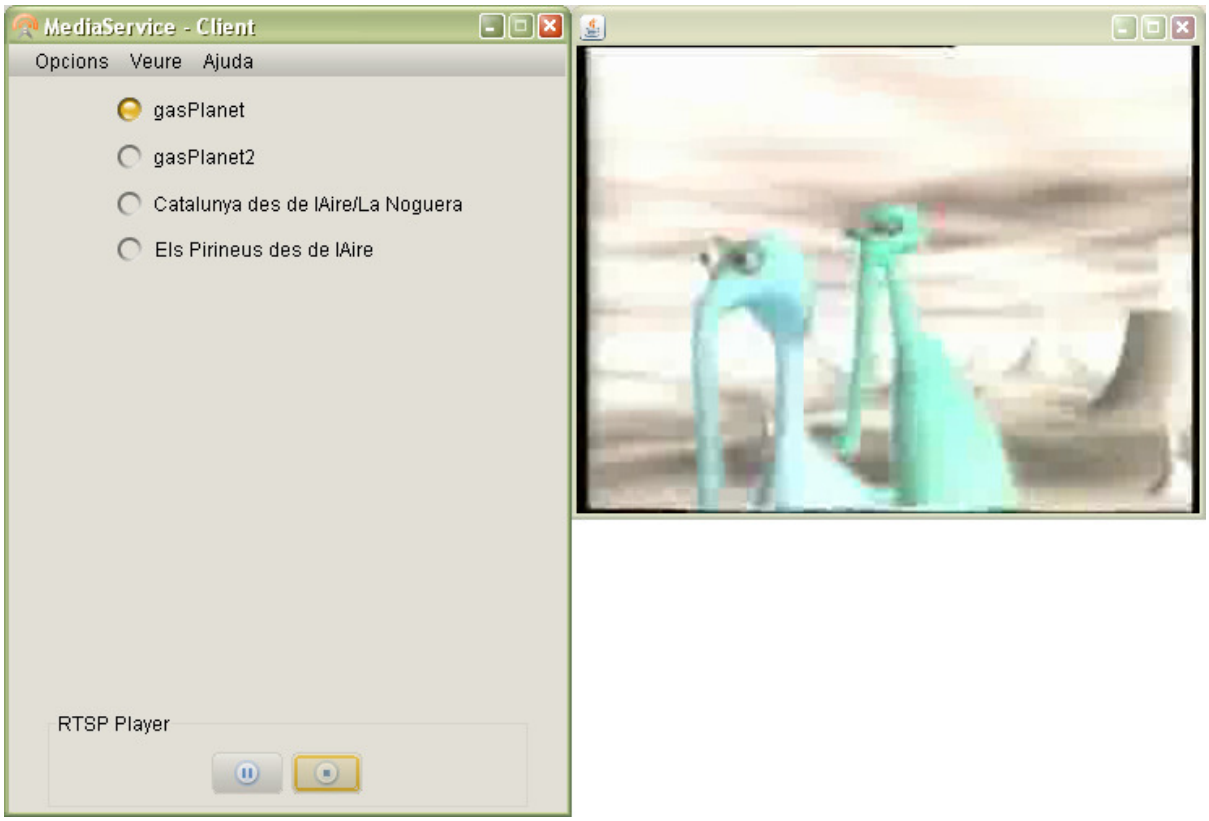

Fig.5.14. Reproducció d'un flux de vídeo

5.6.3. Reproductor RTSP

En el capítol anterior s'ha vist quines eren les funcions d'aquest mòdul: rebre i reproduir un flux multimèdia.

A continuació es presenta quins han estat els elements necessaris per a la seva implementació.

Des d'un principi sempre s'ha tingut la idea de poder integrar un reproductor de vídeo dins d'una aplicació java com el MediaClient.
Després de dies de cerca de possibilitats es va trobar una possible solució: JVLC[20] (Java VideoLan Client)[13].

JVLC és una implementació en java del conegut reproductor de vídeo VideoLan.
Un cop obtingudes les llibreries necessàries, només cal fer-ne ús per crear una aplicació que integri un reproductor de vídeo, concretament, un VideoLan.

Un cop solucionat el problema d'integració del reproductor, va arribar el moment de pensar en el protocol RTSP. Calia, doncs, implementar una sèrie de llibreries per tal de poder treballar amb RTSP. Però la troballa del JVLC no

[20] Documentació JVLC, veure Annex C.

solsament va solucionar la integració d'un reproductor de vídeo, si no que també va evitar que s'hagués d'implementar cap llibreria RTSP.

Parlar de JVLC és com parlar de VideaLan. VideoLan no solsament és un reproductor multimèdia si no que també pot realitzar tasques de client RTSP.

Per tant, amb JVLC s'ha aconseguit integrar, no un simple reproductor multimèdia, sinó un reproductor RTSP.

Tota la integració i implementació del reproductor s'ha realitzat en la classe JPanelRTSPPlayer. Aquest és el seu diagrama de classe:

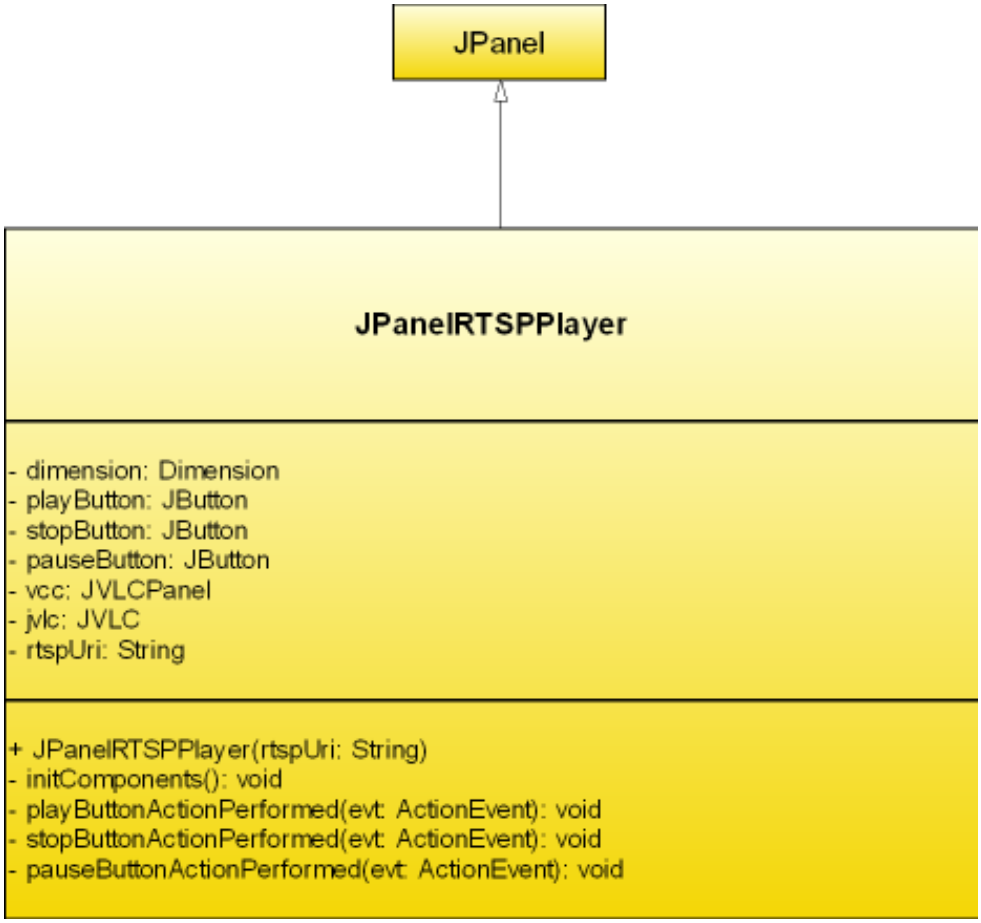

Fig.5.15. Diagrama de classe de JPanelRTSPPlayer

Aquesta classe forma part del conjunt de classes que conformen la interfície gràfica.

Gràcies a la funció del MediaClient Controller d'obtenir la localització d'un recurs, el reproductor RTSP serà capaç de dialogar amb el servidor RTSP que conté el recurs multimèdia, i posteriorment rebre i reproduir el flux multimèdia.

En JVLC continua existint un dels inconvenients del reproductor VideoLan. Quan aquest actua com a client RTSP, no té la capacitat de suportar la comanda RTSP PAUSE. Per tant, no és capaç d'aturar temporalment la reproducció i recepció del flux multimèdia. En capítols posteriors, s'exposen possibles solucions a aquest problema.

5.7. SIP User Agent

La forma en que tots els elements participants del sistema estableixen comunicació és mitjançant el protocol SIP. Per aquest motiu, i com ja s'ha exposat en capítols anteriors, és necessari que tots els participants, clients i servidor, tinguin un SIP User Agent que els ajudi a establir comunicacions a través d'aquest protocol.

La implementació del SIP User Agent, tant en el MediaClient com en el MediaServer, s'ha realitzat mitjançant una classe de Java anomenada SipUA.
El diagrama de classe de SipUA és el següent:

Fig.5.16. Diagrama de classe del SipUA

L'atribut messageBox, el qual es pot observar en aquest diagrama, és el que utilitzarà el SipUA per entregar els missatges rebuts a altres mòduls presents el les diferents implementacions.

La classe SipUA fa ús de la llibreria JAIN-SIP[21].
JAIN-SIP[8] és una interfície java que permet la creació d'aplicacions que requereixen del protocol SIP. Aquesta interfície estandarditza la interfície de la pila SIP, els missatges SIP i el tractament d'events com la recepció d'un missatge SIP.

5.8. Passarel·la SIP-RTSP

Aquest element forma part del projecte *Machine* desenvolupat per i2Cat.
Durant la realització d'aquest treball s'ha donat suport en el seu desenvolupament.
Tal i com s'ha anat comentant al llarg d'aquest document, alguns dels serveis desenvolupats en aquest treball, poden ser integrats en el projecte *Machine*. Per aquests dos motius es descriu a grans trets la implementació de la Passarel·la o Gateway SIP-RTSP.

Aquest element permet, a tavés de trucades IP, controlar i consumir recursos multimèdia presents en la xarxa.

En el primer capítol s'ha vist quins eren els elements presents en la arquitectura de la passarel·la. Per implementar aquests elements s'està fent ús de programari adient per a la realització de les diferent funcions.
Aquest programari és el següent:

- Client SIP o SIP User Agent: **X-Lite - SIP softphone**[24]
- Gateway SIP-RTSP: **Asterisk**[25]
- Servidor RTSP: **Fenice RTSP Server**[11]

X-Lite – SIP softphone és una aplicació que permet l'inici de sessions SIP per a la senyalització de trucades telefòniques a través de la xarxa. Aquest software incorpora una interfície gràfica força senzilla, cosa que permet a l'usuari interactuar còmodament amb el programa.

[21] JAIN-SIP http://jain-sip.dev.java.net , veure documentació a Annex C

Fig.5.17. Interfície gràfica de X-Lite

Asterisk és un programari que implementa les funcions d'una central telefònica o PBX. Aquest permet la Resposta Interactiva per Veu (IVR). Això vol dir que és capaç de reconèixer comandes per veu. Aquesta característica permet enviar comandes a través d'una trucada.
A més Asterisk permet interactuar amb aplicacions externes, fet que permet, juntament amb la funció de IVR, oferir serveis a través d'una trucada.

El que s'aconsegueix amb Asterisk és controlar, amb l'ajut d'un client RTSP (desenvolupat en aquest treball amb l'ajut de JVLC), una sessió multimèdia a través de comandes de veu.

Asterisk, per sí mateix, no pot executar codi extern. Però la incorporació de **l'Asterisk Gateway Interface** o **AGI** permet a l'Asterisk fer ús de programari extern. L'AGI és una interfície que permet l'addició a l'Asterisk de noves funcionalitats programades en diferents llenguatges de programació (Java, Perl, C, PHP, etc). Gràcies a la incorporació de AGI es pot utilitzar el client RTSP dissenyat i implementat en aquest treball[22].

[22] Client RTSP, veure apartat 5.6.3.Reproductor RTSP.

L'esquema definitiu de tota la arquitectura és el següent:

Fig.5.18. Arquitectura del sistema Gateway SIP-RTSP

5.9. Futures implementacions

En el moment de finalitzar la implementació, el resultat final cobria totes les necessitats que en un principi es van plantejar:

- Distribució d'informació de recursos multimèdia. Realitzat pel MediaServer.
- Recepció i reproducció dels recursos. Realitzat per l'RTSPPlayer en el MediaClient.
- Mobilitat dels participants. Implementat amb l'ús del protocol SIP i tots els elements que l'implementen.

Tot i tenint un resultat satisfactori, es poden incloure millores i noves característiques.

5.9.1. Compatibilitat de la comanda RTSP PAUSE en JVLC

Durant la implementació d'un dels elements del sistema, concretament el reproductor RTSP, es va topar amb un inconvenient: JVLC no suporta la comanda PAUSE de RTSP.

Gràcies al suport que s'ha donat en el desenvolupament del projecte Machine s'ha trobat una solució a aquest problema. S'ha realitzat la implementació de llibreries RTSP i d'elements necessaris per dur a terme tota la lògica de funcionament. Ara, doncs, l'ús que es fa del JVLC no és de client RTSP si no de reproductor multimèdia. Això és degut a que el JVLC ja no controlarà les sessions RTSP i només s'encarregarà de reproduir els fluxos multimèdia que es rebin dels servidors RTSP.

5.9.2. Noves característiques

Fita de cara a treballs futurs, és el disseny i implementació de noves característiques en els elements. A continuació s'exposen algunes d'elles:

- **RTSP Server mòbil:**
  De la mateixa forma que el MediaServer i MediaClient tenen la possibilitat de canviar de localització dins la xarxa sense perdre la seva visibilitat en el domini, es pot incorporar aquesta característica al RTSP Server.
  Per incorporar aquesta característica només cal introduir l'RTSP Server dins del domini SIP. La incorporació d'un SIP User Agent permetria la entrada d'aquest element dins del domini.

- **Comunicacions segures:**
  Actualment totes les comunicacions SIP viatgen per la xarxa en clar, qualsevol usuari podria visualitzar el contingut de tots els missatges. Una solució passaria per xifrar tots els missatges SIP.
  Per a que el sistema tingui aquesta característica cal modificar les implementacions del SIP User Agent i incloure-li aquesta funció.
  Pel que fa al SIP Proxy Server no presentaria cap inconvenient, ja que el SER suporta aquesta opció.

5.10. Programari utilitzat

Per a la implementació del mediaServer i mediaClient s'ha fet ús de divers tipus de software. En aquest apartat es llisten tots i cada un dels programes utilitzats.

5.10.1. NetBeans 5.5

Per al disseny i implementació de les classes de Java descrites en apartats anteriors, s'ha fet ús de la interfície de desenvolupament NetBeans 5.5.

NetBeans és una interfície de desenvolupament d'aplicacions escrites en Java.
Aquesta aplicació porta incorporada una sèrie d'eines que faciliten el disseny i implementació de qualsevol aplicació basa en Java.
Permet la compilació i execució del codi font implementat.
També permet la creació de diagrames UML necessaris per ajudar en el disseny de les aplicacions.
Ajuda al programador a introduir la sintaxi correcta del llenguatge de programació.

Es poden trobar altres característiques a la pàgina oficial de NetBeans[22]

5.10.2. MySQL GUI Tools

En la creació i gestió de la base de dades del MediaServer s'han utilitzat algunes de les eines proveïdes per MySQL GUI Tools.

MySQL GUI Tools conté totes les eines necessàries per la gestió de base de dades MySQL. Les eines utilitzades són:

- **MySQL QueryBrowser**. Aplicació que permet la creació, modificació i consulta a base de dades mysql.

- **MySQL Administrator**. Eina d'administració i gestió de la base de dades mysql.

Aquestes i altres característiques es poden consultar a la pàgina web oficial de MySQL[21].

5.11. Altre Programari

En la implementació del servidor RTSP es van estudiar i provar varis programes que realitzaven funcions de servidor RTSP. Aquests van ser: Fenice, Darwin Streaming Server[15] i Helix Server[16].

Com s'ha comentat en l'apartat d'implementació del RTSPServer, es va decidir per Fenice d'entre els tres perquè permet l'ús de recursos codificats en els estàndards més comuns (MPEG-1/2 i MPEG-4) i a més suporta la incorporació de serveis de LiveStreaming.

Amb els altres dos programes es van trobar alguns inconvenients:

- Només permeten la entrega de recursos codificats amb códecs propietari. Tant en Darwin com en Helix calia la recodificació dels fitxers amb els códecs propis de cada casa.
- Com a clients RTSP només eren compatibles els propis de cada empresa: QuickTime, en el cas de Darwin, i RealPlayer per a Helix.

L'únic avantatge que presentaven era que, a diferència amb Fenice, presentaven una interfície gràfica força amigable i senzilla d'utilitzar.

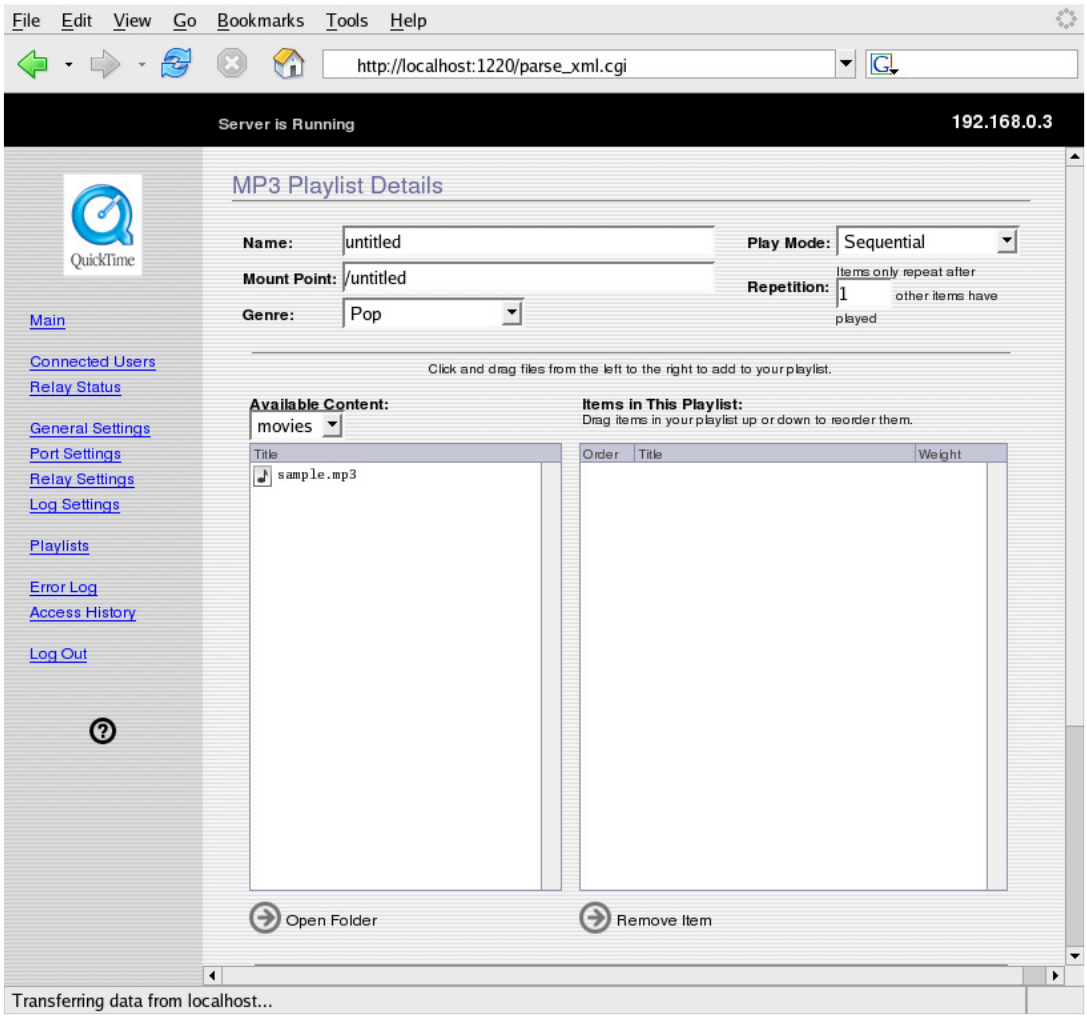

Fig.4.19. Interfície gràfica del Darwin Streaming Server

CAPÍTOL 6. PLANIFICACIÓ

Per a la correcta realització i control del treball es va realitzar la planificació que a continuació es presenta.

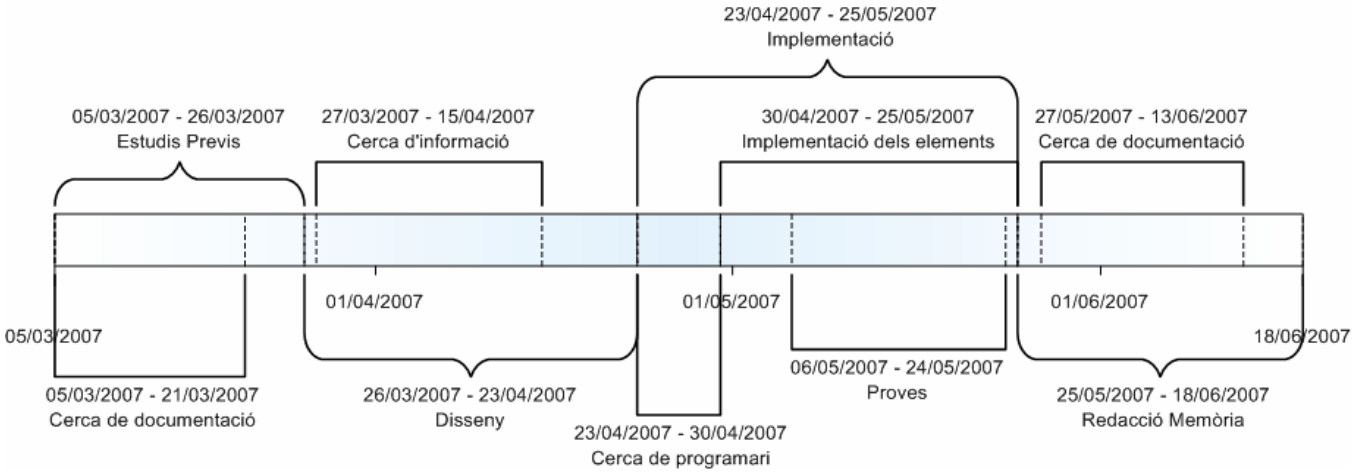

Fig. 6.1. Eix temporal del desenvolupament del treball.

El treball es va dividir en 4 apartats:

- **Estudi**: Temps dedicat a l'adquisició i assimilació de coneixements.
- **Disseny**: Temps per la definició de les funcions i característiques del projecte.
- **Implementació**: Dedicació per a la implementació del disseny previ.
- **Redacció**: redacció de la memòria sobre el desenvolupament de tot el projecte.

En la taula següent es presenten quines han estat les tasques realitzades i els temps estimats i dedicats a cada una d'elles:

Taula 6.1. Tasques i temps de dedicació

Apartat	Tasca	Descripció	Temps estimat	Temps dedicat
Estudi	Estudi de protocols	Estudi dels protocols SIP i RTSP	30	30
Estudi	Cerca d'implementacions	Estudi de les possibilitats d'implementació de cada protocol.	10	15
Estudi	Situacions	Estudi de les situacions i casos d'implementació dels protocols	8	8
Estudi	Proposta de situació	Proposta d'un escenari on l'aplicació dels dos protocols ajudaria a la seva implementació	2	2
Disseny	Especificació dels elements	Especificació del nombre d'elements i funcions necessaris per cobrir les necessitats	10	10
Disseny	Definició i disseny del servidor RTSP	Disseny i especificació de les funcions de l'RTSP Server	10	20
Disseny	Disseny del servidor d'informació	Disseny i especificació del MediaServer	45	50
Disseny	Disseny del client	Disseny i especificació del MediaClient	45	50
Implementació	Cerca de programari	Cerca de programari adient per a la implementació dels elements	30	30
Implementació	Implementació dels elements	Implementació de cada element.	120	130
Implementació	Proves	Proves de funcionament de cada element	20	30
Implementació	Posada en funcionament	Arrencada de tots els elements.	1	2
Redacció	Memòria	Redacció de la memòria sobre el projecte desenvolupat	100	110

El temps total de dedicació ha estat de 487 hores. Tenint en compte que el temps de dedicació setmanal ha estat d'una mitja de 30 hores, el nombre de setmanes per a l'adquisició de coneixements, disseny i desenvolupament del projecte ha estat de 16 setmanes.

La diferència d'hores entre les estimades i dedicades és de 56, unes dues setmanes. Els motius entre altres han estat: problemes de compatibilitat d'alguns dels programes trobats per a realitzar les implementacions, falta de previsió dels coneixements de programació alhora de realitzar la implementació d'interfícies gràfiques, cerca de solucions a problemes com la incompatibilitat del JVLC amb la comanda RTSP PAUSE.

CAPÍTOL 7. CONCLUSIONS

A continuació es presenten les conclusions extretes de la realització del projecte. Les conclusions estan diferenciades per: objectius aconseguits, objectius per al futur, conclusions personals i impacte mediambiental que presenta el desenvolupament del projecte.

7.1. Objectius

A l'inici del procés de disseny del sistema es van definir uns objectius a assolir:

- Distribució de recursos multimèdia. Servei de Vídeo sota Demanda.
- Donar servei de fluxos en viu o LiveStreaming
- Mobilitat dels participants: client i servidor.
- Incorporació del servei en el Paquet de Treball de Videoconferència del projecte Machine de l'i2Cat.

Tots els objectius han estat assolits en el moment de finalitzar el desenvolupament del treball.

La incorporació d'alguns elements presents en el treball al projecte *Machine*, en el moment de redacció d'aquest document, no ha estat duta a terme. El motiu és que en aquest moment el projecte no està suficientment desenvolupat com per començar a incorporar els serveis aquí descrits.

7.2. Futurs objectius

Un cop assolits els principals objectius, es hora de començar a pensar en altres fites. En un futur es poden incorporar noves funcions i característiques al sistema com les que es presenten a continuació:

La possibilitat d'automatitzar la inclusió d'informació en la base de dades en el moment d'incloure nous recursos, podria ser ara per ara una de les vies a seguir.

Un altre camí a seguir és el desenvolupament d'un mòdul que permeti distribuir la informació de la base de dades a altres maquines i permetre balanceig de càrrega d'informació.

Incloure els servidors RTSP dins del domini SIP i per tant permetre la seva mobilitat, també és un objectiu a aconseguir.

Aquestes i altres idees poder servir per acabar de desenvolupar un sistema que, en un futur, podria ser aplicat comercialment com una nova via de divulgació audiovisual.

7.3. Conclusions Personals

La realització d'aquest treball m'ha permès l'adquisició de nous coneixements i noves metodologies de treball.
Desenvolupar un servei des del plantejament de la problemàtica fins a la posada en marxa d'aquest ha estat enriquidor i m'ha donat la motivació per continuar en aquesta línia: la investigació i desenvolupament de nous serveis telemàtics.

La possibilitat d'integrar el servei en un projecte en desenvolupament com el *Machine*, permet veure que el treball desenvolupat té una aplicació pràctica i útil.

7.4. Impacte Mediambiental

Aquest projecte, mediambientalment, no presenta, a priori, cap impacte. L'únic que podria perjudicar és el procés de fabricació i destrucció futura de les màquines utilitzades, com també el desgast energètic durant la realització i posada en funcionament del mateix.

Es podria dir, que el fet de divulgar continguts per la xarxa (una infraestructura ja desenvolupada i en funcionament), evita que s'hagi d'utilitzar suports físics d'emmagatzematge (USB Memory, CDs, DVDs ...) per realitzar la mateixa funció de distribució.
Evitar l'ús de CDs i DVDs implica evitar l'ús de productes i plàstics procedents de recursos fòssils com el policarbonat. A més, el reciclatge d'aquests components és difícil i costós.

Es podria dir, doncs, que el desenvolupament d'aquest projecte és mediambientalment sostenible.

BIBLIOGRAFIA

[1]- **SIP**: RFC 3261 referent al Session Initiation Protocol [última consulta: 25 de maig del 2007]
URL: http://www.ietf.org/rfc/rfc3261.txt

[2]- **SIP Center**: Portal de referència per al desenvolupament d'aplicacions comercials de SIP [última consulta: 25 de maig del 2007].
URL: http://www.sipcenter.com/

[3]- **SIP Tutorial**: PDF amb referències històriques i descripció de l'ús de SIP [útlima consulta: 27 de maig del 2007]
URL: http://www.iptel.org/files/sip_tutorial.pdf

[4]- **Understanding SIP**: PDF on es descriu el protocol SIP, els seus elements i les seves possibilitats. [última consulta: 27 de maig del 2007]
URL:
http://www.sipcenter.com/sip.nsf/html/WEBB5YNVK8/$FILE/Ubiquity_SIP_Overview.pdf

[5]- **RTSP**: RFC 2326 referent al Real Time Streaming Protocol. [última consulta: 28 de maig del 2007]
URL: http://www.ietf.org/rfc/rfc2326.txt

[6]- **Internet Media on Demand**: PDF explicatiu de RTSP i les seves aplicacions. [última consulta: 26 de maig del 2007]
URL: http://www.cs.columbia.edu/~hgs/teaching/ais/slides/2003/RTSP.pdf

[7]- **SER**: web oficial del projecte SIP Express Router. [última consulta: 30 de maig del 2007].
URL: http://www.iptel.org/ser/

[8]- **JAIN-SIP**: web oficial del projecte JAIN-SIP. [última consulta: 1 de juny del 2007]
URL: https://jain-sip.dev.java.net/

[9]- **JAIN-SIP Javadoc**: documentació oficial de la llibreria JAIN-SIP. [última consulta: 20 de maig del 2007]
URL: http://snad.ncsl.nist.gov/proj/iptel/jain-sip-1.2/javadoc/

[10]- **JAIN-SIP Tutorial**: article descriptiu de les funcionalitat de la llibreria JAIN-SIP [última consulta: 14 de juny del 2007]
URL: http://java.sun.com/products/jain/JAIN-SIP-Tutorial.pdf

[11]- **Fenice**: web oficial del servidor RTSP [última consulta: 1 de juny del 2007]
URL: http://streaming.polito.it/legacy_server

[12]- **VideoLan**: web oficial del projecte VideoLan [última consulta: 22 de maig del 2007]
URL: http://www.videolan.org/

[13]- **Java VLC**: web oficial del projecte JVLC. [última consulta: 4 de juny del 2007]
URL: http://trac.videolan.org/jvlc

[14]- **Java Platform SE 6**: API de la especificació 6 de Java. [última consulta: 5 de juny del 2007]
URL: http://java.sun.com/javase/6/docs/api/

[15]- **Darwing Streaming Server**: web oficial del projecte del servidor RTSP Darwing. [última consulta: 25 d'abril del 2007]
URL: http://developer.apple.com/opensource/server/streaming/index.html

[16]- **Helix Server**: web oficial de Real Networks sobre el servidor Helix. [última consulta: 25 d'abril del 2007]
URL: http://www.realnetworks.com/products/media_delivery.html

[17]- **Ubuntu**: pàgina oficial amb tota la documentació referent a aquesta distribució Linux. [última consulta: 15 de maig del 2007]
URL: http://www.ubuntu.com

[18]- **FFMPEG**: web oficial del projecte del multicodificador ffmpeg. [última consulta: 2 de juny del 2007]
URL: http://ffmpeg.mplayerhq.hu/

[19]- **LAME MP3 Encoder**: web oficial del projecte LAME, on part del projecte és el desenvolupament d'un codificador MP3 [última consulta: 2 de juny del 2007]
URL: http://lame.sourceforge.net/index.php

[20]- **MPLAYER**: web oficial del reproductor multimèdia MPlayer. [última consulta: 2 de juny del 2007]
URL: http://www.mplayerhq.hu/

[21]- **MySQL**: web oficial [última consulta: 3 de juny del 2007]
URL: http://www.mysql.com

[22]- **NetBeans**: web de la interfície de desenvolupament java NetBeans. [última consulta: 5 d'abril del 2007]
URL: http://www.netbeans.org/

[23]- **i2Cat**: web de la fundació i2Cat. [última consulta: 10 de juny del 2007]
URL: http://www.i2cat.cat

[24]- **X-Lite - SIP softphone**: web descriptiva del client SIP X-Lite. [última consulta: 15 de juny del 2007]
URL: http://www.asteriskguru.com/tutorials/xlite_softphone.html

[25]- **Asterisk**: web oficial del projecte. [última consulta: 16 de juny del 2007]
URL: http://www.asterisk.org

[26]- **An Overview of MPEG-2**: article publicat per Hewlett Packard sobre l'estàndard de codificació MPEG-2. [última consulta: 18 de juny del 2007]
URL: http://cp.literature.agilent.com/litweb/pdf/5966-1031E.pdf

Com a punt de partida de cerca d'informació i per tenir una primera idea de molts dels conceptes s'ha utilitzat informació de la enciclopèdia lliure **Wikipedia**.
URL: http://en.wikipedia.org

ANNEXOS

ANNEX A. CONFIGURACIONS I INSTAL·LACIÓ

A continuació es descriuen tots els passos necessaris per configura i instal·lar el programari utilitzat per muntar i posar en funcionament el sistema.

Requeriments mínims per al SIP Proxy Server i RTSP Server:

- Pentium III 1Ghz
- 254MB de memòria RAM
- Targeta de xarxa Fast Ethernet
- 5GB de disc dur
- Ubuntu Linux 6.06LTS

Requeriments mínims per al MediaServer i MediaClient:

- Pentium III 1Ghz
- 512MB de memòria RAM
- Targeta de xarxa Fast Ethernet
- 10GB de disc dur
- Windows XP
- Màquina Virtual de Java (JVM http://java.sun.com)

A.1. Configuració i Instal·lació del SIP Express Router en el SIP Proxy Server

La distribució Ubuntu té la possibilitat d'instal·lar el SER des dels seus repositoris. Només cal executar la comanda:

```
$ apt-get install ser
```

La configuració per defecte del SER un cop instal·lat és la adient per a l'ús que se'n farà.

Per executar-lo caldrà executar la següent comanda:

```
$ ser -d
```

A.2. Configuració i instal·lació del Fenice i LiveStreaming en el RTSP Server

Previ a la instal·lació cal obtenir el codi font de la web oficial de Fenice:
http://streaming.polito.it/legacy_server

La configuració i instal·lació de Fenice és la següent:

1.- Desempaquetar el paquet descarregat:

```
$ tar xvfz fenice-1.12.tar.gz
```

2.- Accedir al directori on s'ha desempaquetat:

```
$ cd fenice-1.12
```

3.- Compilar el codi:

```
$ ./configure && make
$ su -
```

4.- Instal·lar

```
# make install
```

El Fenice es troba instal·lat.

Per defecte la carpeta on el fenice considera que té els fitxers de descripció dels recursos multimèdia és */var/fenice/avroot*.

Aquesta configuració es pot modificar canviant els paràmetres del fitxer de configuració */etc/fenice.conf*

Aquest fitxer presenta la següent estructura:

```
------------------------------------------------------------
# Define root dir where media are stored
root=/var/fenice/avroot
# Define RTSP port where Fenice listens for new connections
port=8554
# Define max session (calculate it according to your bandwith)
max_session=10
#------------------------------------------------------------
```

Ara el Fenice es troba preparat per ser executat:

```
$ fenice --config-file /etc/fenice.conf
```

Cada nou recurs que s'inclogui en el servidor RTSP cal que vagi acompanyat d'un fitxer de tipus SD. Aquest fitxer ha de contenir tota la informació necessària sobre el recurs.
A continuació es presenten dos fitxers SD, un d'un fitxer d'àudio i l'altre de vídeo:

```
stream
        default_presentation
        priority 1
        file_name audio.mp3
        media_source stored
        payload_type 14
        clock_rate 90000
        audio_channels 2
        encoding_name MPA
        sample_rate 44100
        coding_type frame
```

```
        frame_len 26.12
    stream_end

    stream
        file_name video.mpg
        priority 1
        media_source stored
        payload_type 32
        clock_rate 90000
        encoding_name MPV
        coding_type frame
        frame_rate 25
        byte_per_pckt 1900
    stream_end
```

Per incorporar la funció de LiveStreming cal seguir les següents passes:

1.- Descarregar ffmpeg de :
http://www.mplayerhq.hu

Ffmpeg està incorporat en el reproductor MPlayer el qual es pot descarregar de la plana web indicada a dalt.

2.- Instal·lar:

```
$ ./configure
$ make
$ make install
```

3.- Crear un fitxer FIFO:

```
$ mkfifo –m 655 pipeaudio.mp3
```

4.- Executar ffmpeg:

```
$ ffmpeg –y –ad /dev/dsp –ar 44100 –ac 2 –vn –ab 128 pipeaudio.mp3
```

5.- Iniciar un reproductor multimèdia com, per exemple, VideoLan i crear una llista de reproducció.

Ara cal incloure un fitxer SD per al flux de LiveStreaming:

```
    stream
        default_presentation
        priority 1
        file_name pipeaudio.mp3
        media_source live
        payload_type 14
        clock_rate 90000
        audio_channels 2
        encoding_name MPA
        sample_rate 44100
        coding_type frame
```

```
        frame_len 26.12
    stream_end
```

Realitzades totes les passes l'RTSP Server ja està en servei.

A.3. Configuració del MediaServer

Per configurar el MediaServer cal modificar el fitxer de configuració serverconfig.PROPERTIES.
Aquest fitxer té la següent estructura:

Fig. A.1. Fitxer de configuració del MediaServer

Un cop desats els canvis ja es pot executar el MediaServer fent doble clic en l'executable.

A.4. Configuració del MediaClient

El MediaClient es pot configurar de la mateixa forma que el MediaServer, mitjançant la edició del fitxer de configuració *clientconfig.txt*

Fig. A.2. Fitxer de configuració del MediaClient

També es pot configurar a través de la interfície gràfica.
Per executar-lo només cal fer doble clic sobre l'executable.

A.5. Transcodificació dels recursos multimèdia

En el cas que es vulgui transcodicar els recursos multimèdia presents en el servidor RTSP es pot fer ús de programes que permeten aquesta funció.
Durant la realització de proves es van realitzar transcodificacions amb l'ajuda de programes com LAME MP3 Encoder i MPLAYER. Tots ells executats des del servidor RTSP el qual està muntat sobre la distribució Linux Ubuntu 7.04.

Les comandes necessàries per a dur a termes les transcodificacions són les següents:

- Fitxers MP3 amb LAME MP3 Encoder:
  $ lame -b [bitrate] --resample [freqüència mostreig] -
  [nomrecurstranscodificat].mp3 < [nomrecursatranscodificar]

- Recursos de vídeo amb MPLAYER i LAME es pot utilitzar el següent script:

```
#--> encoding variables <--#
asr="44.1" # audio sample rate
abitrate="128" # audio bitrate
# vbitrate="2048" # video bitrate
framerate="3"
aspect="2"
if [ -z $1 ]; then
        echo
        echo "$0: You must provide a filename..."
        echo "Exiting."
        echo
        exit 1
fi
vbitrate=`basename  $2` # video bitrate
infile=`basename $1`
outfile=${infile%.*}
```

```
if [ -n "$aspect" ]; then
        aspect="-a $aspect"
fi
if [ -n "$framerate" ]; then
        framerate="-F $framerate"
fi
if [ -n "$vbitrate" ]; then
        vbitrate="-b $vbitrate"
fi
#--> variables <--#
AUDIO="audiodump.wav"
VIDEO="stream.yuv"
# MPLAYER_CMD="mplayer -noframedrop -vo yuv4mpeg -nosound -
v -osdlevel 0"
MPLAYER_CMD="mplayer -noframedrop -vo yuv4mpeg -ao
pcm:waveheader -v -osdlevel 0"
MPEG2ENC_CMD="mpeg2enc -v 0 $aspect $vbitrate -f 0
$framerate -R 2 -c -s -g 10 -G 10 -o"
LAME_CMD="lame -b $abitrate --resample $asr"
# DENOISE="cat -" # OR:
DENOISE="yuvdenoise"

#i=0
#while [ -e $outfile.mpg ] || [ -e $outfile.mp3 ]; do
#        outfile=${infile%.*}_$i
#        let i++
#done
#echo -ne "Outputting:\n\tVideo to $outfile.mpg\n\tAudio to
$outfile.mp3\n"
rm -f $VIDEO
mkfifo -m 660 $VIDEO
echo "launching: $MPLAYER_CMD"
$MPLAYER_CMD $1 &
$DENOISE < $VIDEO | $MPEG2ENC_CMD $outfile.video.mpg
wait
$LAME_CMD - $outfile.audio.mp3 < $AUDIO
rm -f $VIDEO
rm -f $AUDIO
#echo -ne "Outputting:\n\tVideo to $outfile.mpg\n\tAudio to
$outfile.mp3\n"
echo
```

Fig.A.3. Script de transcodificació de recursos de vídeo

ANNEX B. MISSATGES I COMANDES

A part dels missatges corresponents als protocols SIP i RTSP, també existeix l'intercanvi d'altres missatges entre el mediaClient i el mediaServer.
Aquest intercanvi serveix per donar les ordres i respostes referents a funcions internes del sistema.

Tots els missatges són enviats a través de missatges SIP tipus MESSAGE. Dins del cos del missatge s'inclouen els missatges i comandes pròpies de l'aplicació.

Els missatges generats en el client o mediaClient són els següents:

- SEND.VIDEOLIST
- SEND.AUDIOLIST
- SEND.RADIOLIST
- SEND.MEDIAURI

Tots ells generen una resposta per part del servidor o mediaServer:

- RESPONSE.MEDIALIST
- RESPONSE.MEDIAURI

A continuació es presenta una taula descriptiva de tots els missatges:

Taula B.1. Descripció dels missatges intercanviats entre mediaClient i mediaServer

Missatge	SEND.VIDEOLIST
Generat per	mediaClient
Descripció	Indica al mediaServer la voluntat de l'usuari de veure una llista de tots els recursos de vídeo disponibles.
Resposta que genera el receptor	RESPONSE.MEDIALIST + <llistat dels recursos de vídeo>

Missatge	SEND.AUDIOLIST
Generat per	mediaClient
Descripció	Indica al mediaServer la voluntat de l'usuari de veure una llista de tots els recursos d'àudio disponibles.
Resposta que genera el receptor	RESPONSE.MEDIALIST + <llistat dels recursos d'àudio>

Missatge	SEND.RADIOLIST
Generat per	mediaClient
Descripció	Indica al mediaServer la voluntat de l'usuari de veure una llista de tots els recursos de LiveStreaming disponibles.
Resposta que genera el receptor	RESPONSE.MEDIALIST + <llistat dels recursos de LiveStreaming>

Missatge	SEND.MEDIAURI + <idMedia>
Generat per	mediaClient
Descripció	Indica al mediaServer la voluntat de l'usuari de consumir un recurs l'identificador del qual està indicat per <idMedia>
Resposta que genera el receptor	RESPONSE.MEDIAURI + <URI del recurs>

Missatge	RESPONSE.MEDIALIST + <llistat de recursos>
Generat per	mediaServer
Descripció	Indica al mediaClient la entrega de la llista de recursos demanats.
Resposta que genera el receptor	Cap

Missatge	RESPONSE.MEDIAURI + <URI del recurs>
Generat per	mediaServer
Descripció	Indica al mediaClient la URI del recurs sol·licitat
Resposta que genera el receptor	Cap

El llistat dels recursos enviat pel mediaServer té la següent sintaxi:

<idMedia> - <nom del recurs>

On idMedia indica l'identificador únic del recurs.

ANNEX C. DOCUMENTACIÓ I ALTRA INFORMACIÓ

A continuació es presenta documentació de programari extern i altra informació d'interès.

C.1. JVLC

El projecte JVLC està dirigit a permetre l'ús d'una llibreria multimèdia de Java de forma senzilla, flexible i potent. JVLC està construït en base al reproductor multimèdia VideoLan, i per tant té la capacitat de:

- Llegir qualsevol fitxer multimèdia de forma ràpida i eficient
- Transcodificar fitxers multimèdia
- Servir fluxos a través de la xarxa.

A continuació es presenta el diagrama UML de classes de la llibreria:

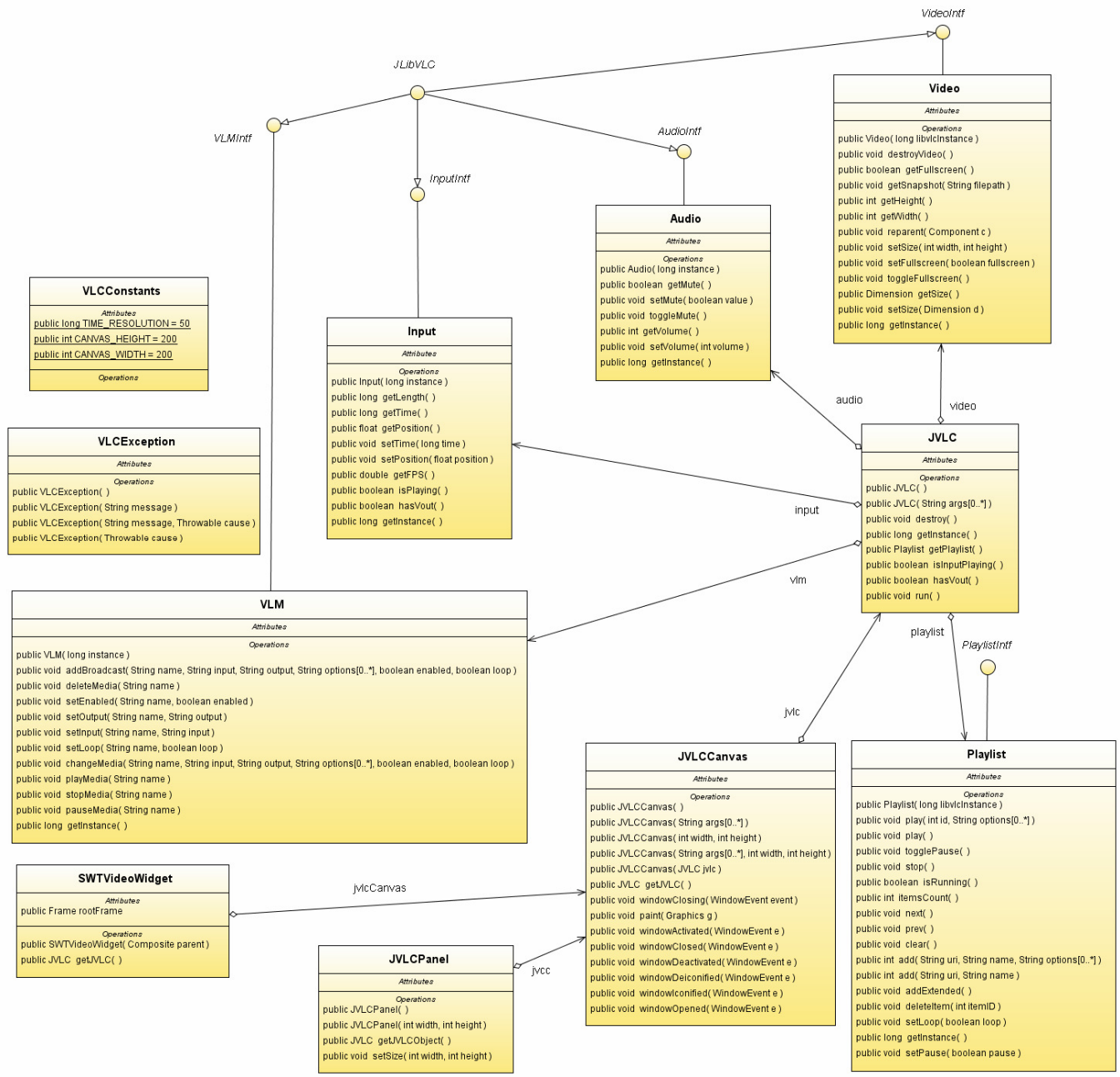

Fig.C.1. Diagrama UML de classes de la llibreria JVLC

C.2. JAIN-SIP

A continuació es presenta un extracte del tutorial publicat per SunMicrosistems sobre la llibreria JAIN-SIP, JAIN-SIP Tutorial .

JAIN-SIP, the **Ja**va-standard **In**terface to a **SIP** signaling stack:

- Standardizes the interface to the stack.
- Standardizes message interface.
- Standardizes events and event semantics.
- Application portability - verified via the TCK.

Designed for developers who require powerful access to the SIP protocol.

JAIN SIP can be utilized in a user agent, proxy, registrar or imbedded into a service container.

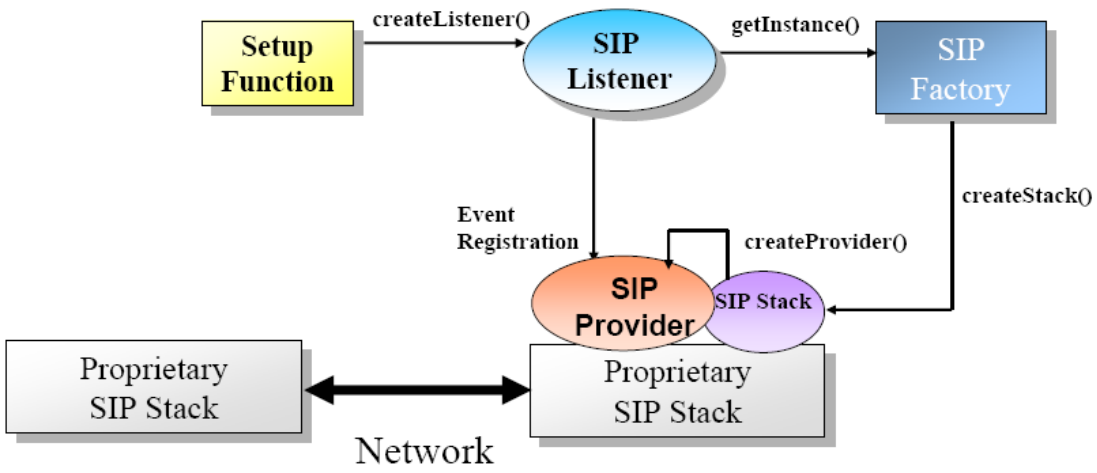

Fig.C.2. JAIN SIP Object Arquitecture

SipStack Interface

Manages Listening Points and Providers.

SipStack associated with an IP address.

- Can have multiple Listening points.

Application can have multiple SipStacks.

Cannot be deleted once created.

Instantiated by the SipFactory and initialized with a property set.

'javax.sip.*' properties are reserved and names defined for stack configuration properties.

Defines retransmission settings.

Defines router information.

Retransmissions

JAIN SIP provides a convenience function that ensures all retransmissions are handled by the JAIN SIP implementation.

- Reduces complexity for applications acting as user agents.
- Reduces complexity for integrating JAIN SIP as a base implementation for a SIP Servlet container or a JAIN SLEE implementation.

Configured via Java properties on the SipStack Interface.

- Default is off.

The default handling of message retransmissions in JAIN SIP is dependent on the application.

- Stateful proxy applications need not be concerned with retransmissions as these are handled by JAIN SIP.

Typically User Agent applications must handle retransmissions of ACK's and 2xx Responses.

SipProvider Interface

Register a SipListener to the SipProvider.

- Notifies registered Listener of Events

De-register a SipListener from the SipProvider.

- Once de-registered, no longer receive Events from SipProvider.

Client and Server Transaction creation methods.

- For sending Request and Response messages statefully.

CallIdHeader creation method.

Send Requests and Responses statelessly.

Listening Point manipulation methods.

- Only one provider per listening point.

Responsibilities of JAIN SIP

Provide methods to format SIP messages.

The ability for an application to send and receive SIP messages.

Parse incoming messages and enable application access to fields via a standardized Java interface.

Invoke appropriate application handlers when protocol significant

- Message arrivals and Transaction time-outs

Provide Transaction support and manage Transaction state and lifetime on behalf of a user application.

Provide Dialog support and manage Dialog state and lifetime on behalf on a user application.

SipListener Interface

A single SipListener per SipStack which implies a single Listener int the architecture.

- All SipProviders associated to a Sipstack have the same SipListener.

Process Request's either statefully or statelessly dependent on application logic.

Process Response's to a recently sent Requests statefully.

Process Transaction timeouts and retransmits Timer events.

- Transaction processing notifications

Responsibilities of the Application

Application registers an implementation of the SipListener interface to interact with the SIP Stack.

Application must register with the SipProvider for all messaging capabilities with the stack.

- Application requests transactions for stateful messaging.
- Application sends stateless messages.
- Access stack objects.

Application receives messages from the stack as Events via the SipListener interface.

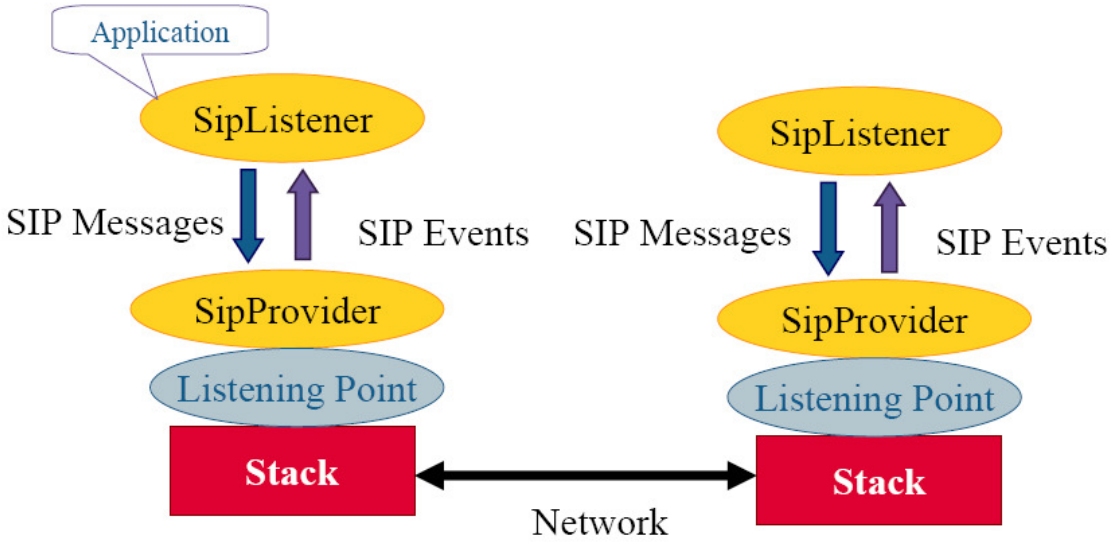

Fig.C.3. JAIN SIP Messaging Architecture

Event Model

The architecture is developed for the J2SE environment therefore is event based utilizing the Listener/Provider event model.

- There is a direct reference between the event provider and event consumer
- Event consumer must register with the event provider

Events encapsulate incoming Requests and Responses.

Event Model is one way i.e. Application doesn't send out events, it sends out messages.

The event model is asynchronous in nature using transactional identifiers to correlate messages.

The SipListener represents the event consumer and listens for incoming Events that encapsulate messages that may be responses to initiated dialogs or new incoming dialogs.

The SipProvider is the event provider who recieves messages from the network and passes them to the application as events.

Packages

General package

- Defines the architectural interfaces, the transaction and dialog interfaces and the event objects of the specification.

Address package

- Address package contains a generic URI wrapper and defines SIP URI and Tel URIs interfaces.

Message package

- Defines the interfaces necessary for the Request and Response messages.

Header packages

- Header package defines interfaces for all the supported headers and extension headers

Factories

JAIN SIP defines four different factories each with respective responsibilities, namely:

- SipFactory
  - This interface defines methods to create new Stack objects and other factory objects.

- AddressFactory
  - This interface defines methods to create SipURI's and TelURL's.

- HeaderFactory
  - This interface defines methods to create new Headers objects.

- MessageFactory
  - This interface defines methods to create new Request and Response objects.

Messages

There are two type of messages in SIP, which JAIN SIP defines as Interfaces:

- Request messages are sent from the client to server.

They contain a specific method type that identifies the type of Request.

A Request-URI which indicates the user or service to which this request is being addressed.

- Response messages are sent from server to client in response to a Request.

They contain a specific status code that identifies the type of Response.
A Request-URI which indicates the user or service to which this request is being addressed.

A reason phrase that is intended for the human user.

Messages may contain multiple Headers of the same type.

- The Headers of a given type within a message is significant, i.e. Headers which are hop-by-hop must appear before any Headers which are end-to-end.

A Message Body may contain a session description.

- JAIN SIP defines this format an Object which allows the body to be a String or an Object type defined the Session Description Protocol (SDP) JSR specification and also a byte array.

Request Message Types

The following request messages are defined by the core SIP protocol:

INVITE

- Invites a participant to a session

BYE

- Ends a client's participation in a session

CANCEL

- Terminates a transaction

OPTIONS

- Queries a participant about their media capabilities

ACK

- For reliability and call accepta nce (3-way handshake)

REGISTER

- Informs a SIP server about the location of a user

The following request messages are defined by various SIP extensions:

INFO

- Session related control information generated during a session.

PRACK

- For reliability of provisional responses.

UPDATE

- Update a session without impacting the state of a dialog.

SUBSCRIBE

- Request notification from remote nodes when certain events occur.

NOTIFY
- Notification from remote nodes when certain events occur.

MESSAGE
- For sending instant messages.

REFER
- Refer to a resource provided in the request.

Headers

SIP headers are similar to HTTP headers fields in both syntax and semantics.

JAIN SIP models each SIP header as a specific interface as opposed to have a single generic interface to handle all header information.
- Each interface specifies the Headers acceptable parameters.
- More explicit protocol support – parsing support for each header.

JAIN SIP supports all the headers defined in RFC 3261 and other headers introduced by supporting the following additional RFC's:

- RFC3262 - RAckHeader and RSeqHeaders for the reliable delivery of provisional responses.
- RFC3265 - AllowEventsHeader, EventHeader and SubscriptionStateHeader to support the event notification framework.
- RFC3326 - ReasonHeader to support information on why the request was issued.
- RFC3515 - ReferToHeader to support recipients to refer requests to another resource

JAIN SIP Extensible by Design

SIP Extensions described in internet drafts and RFCs typically define:
- New SIP Methods

New dialog creating methods
- New SIP Headers.

JAIN SIP defines an extensible framework to support new headers standardized for SIP:

- New SIP methods can be set using the string method field of a request.

- An application informs the stack of dialog creating methods, by specifying the method name to the EXTENSION_METHOD property of the SipStack configuration.

JAIN SIP defines an extensible framework to support new headers standardized for SIP:

- Defines a ExtensionHeader interface that contains the header name and header value attribute pair.
- Can be created and accessed by name.

Transactions and Dialogs

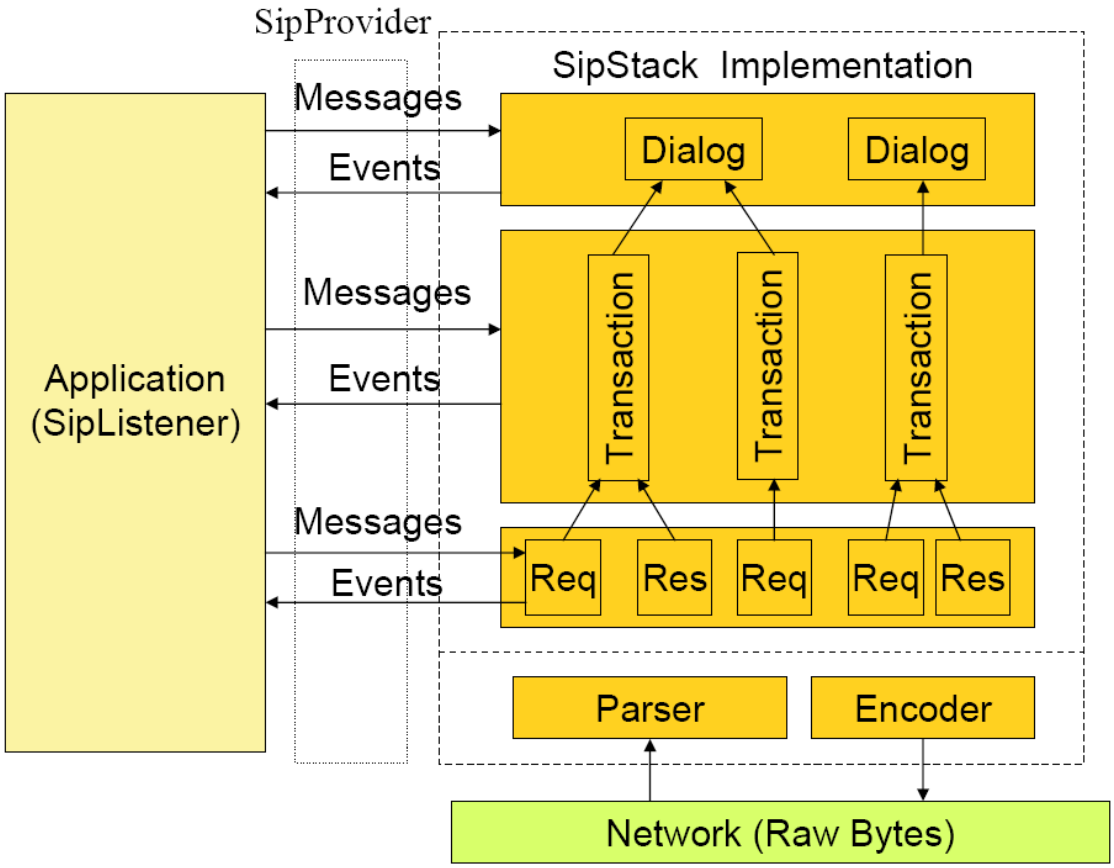

Fig.C.3. Generic SIP Application Structure

SIP Transactions

A SIP transaction consists of a single request and any responses to that request.

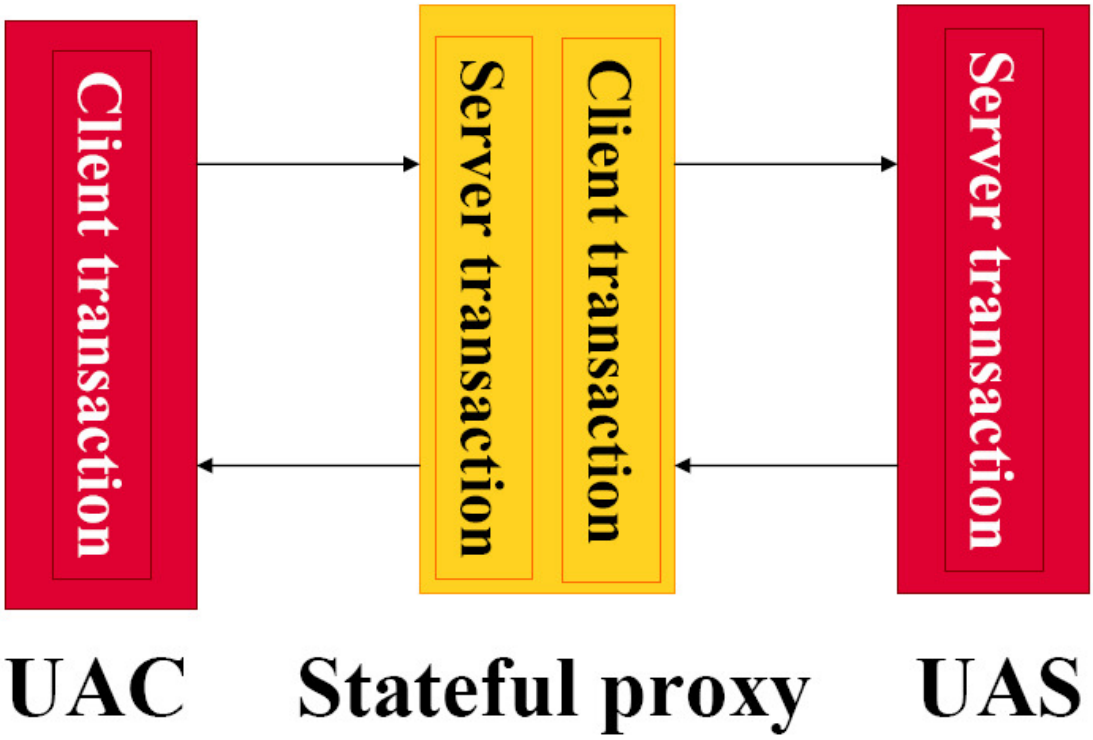

Fig.C.4. SIP Transactions

Transaction Support

JAIN SIP standardizes the interface to the generic transactional model defined by the SIP protocol

- JAIN SIP models both Client and Server Transactions as Interfaces.

Transaction is created on incoming Request or may be created to send outgoing request.

- When a Request is sent out statefully, application must request a ClientTransaction
- When a new Request arrives, application determines whether to handle request via a ServerTransaction
- When a Request in an existing dialog arrives the stack automatically associates it to a ServerTransaction

When a response arrives, the Stack possibly associates a previously created ClientTransaction with the response

- May be stray

Messages are passed to the SipProvider in order to generate a new transaction. This transaction can be used to send the message onto the network

Implementation manages the association between Transactions and Dialogs.

Dialog Support:

A Dialog is a peer to peer association between communicating SIP endpoints.

- The dialog represents a context in which to interpret SIP messages.

Dialogs are never directly created by the Application.

- Dialogs are established by Dialog creating Transactions (INVITE, SUBSCRIBE…) and are managed by the stack.

Dialog deletion may be under application control.

- Though not generally recommended.

Dialogs are used to maintain data needed for further message transmissions within the dialog

- Route Sets, Sequence Numbers, URI's of the parties in the dialog.

Dialogs have a state machine

- Early, Confirmed, Completed and Terminated.

Transactions may belong to a Dialog

- Dialog state changes as a result of changes in Transaction State.
- Access to dialog functionality from the transaction interface.

C.3. MPEG

A continuació es presenta un extracte de l'article An Overview of MPEG-2 [26] on es parla sobre l'estàndard de codificació MPEG.

MPEG stands for Moving Picture Experts Group.
The name was given to the International Standards Organization (ISO) committee that specified a standard compression, transmission, and decompression scheme for video. Today MPEG refers to a series of standards documents officially known as ISO/IEC 11172 (MPEG-1)and ISO/IEC 13818 (MPEG-2). MPEG actually specifies the syntax of the compressed video and audio bitstreams, which enables a great deal of flexibility at the encoder.

The basic job of MPEG is to take analog or digital video signals and convert them into packets of digital information that are more efficiently transported on modern networks. MPEG compresses the video into much less information, consuming less transmission bandwidth-only one-sixth to one-thirtieth of the capacity is needed. A digital transmission, MPEG maintains the transmission quality end to end. As progressively longer transmission networks are used, the signal does not degrade and the picture does not get fuzzy.
There are many important ramifications of the technologies incorporated into the MPEG specifications, but what seems to get the most press is the video compression system. MPEG shares much of its compression technique with a related standard called JPEG. JPEG stands for Joint Pictures Experts Group, hich is another work-group under the same subcommittee (29) of the Joint Technical Committee 1 of the International Electrical Commission of the International Standards Organization. That name is quite a mouth-full so it is commonly referred to as JPEG.
JPEG, a standard for compression of still images, takes advantage of the optical characteristics of the human eye and removes picture information that is not very visible to viewers. The image is stored in a format that removes information, such as high frequency color transitions, which is not readily discerned by the human eye. The eye is much more sensitive to high frequency luminance (brightness) changes than to color changes, due to its rod (luminance) and cone (color) structure. So when the information is converted back into a picture, it is hoped that the human eye does not notice any change in the picture caused by the loss of information in this compression system. The information is lost in the Discrete Cosine Transform, quantization, and entropy coding processes used for the compression.

MPEG takes advantage of spatial and temporal redundancies in video material to compress the information.

The original MPEG standard, MPEG-1 was targeted at producing a trade-off between compression, ease of implementation, and processing power required for decompression. MPEG was designed for broadcast and multimedia (i.e. CD-ROM video) applications. Due to the motion vector searching, the encoding effort for MPEG is asymmetrical-it is more work and requires more expensive, powerful hardware for encoding than decoding. This design was justified

because a video signal will need to be decoded many times for viewers, while it only needs to be encoded once. Optimizing the cost of the decoder for mass production was a key concern. MPEG-1 was designed for optimum efficiency at the bandwidths available on CD-ROM drives (~1.5 Mb/s) and with a restricted degree of complexity.

MPEG-2 is the successor to MPEG-1. MPEG-2 was optimized for the digital compression of TV broadcast material, and yields little visible degradation in quality from CCIR-601 when transmitted at 1.5 to 6 Mb/s.