-TFC Enginyeria Informàtica-

DESENVOLUPAMENT DE SISTEMES INTEGRATS DE VEUIP I MISSATGERIA INSTÀNEA

Autor: Carles Miralles Pena
Director: Victor Carceler Hontoria

ÍNDEX

CAPÍTOL 0 – Presentació

PRESENTACIÓ

El present document conté la memòria del treball final de carrera de l'enginyeria superior en informàtica de l'Universitat Oberta de Catalunya realitzada per l'estudiant Carles Miralles Pena, la qual s'ha desenvolupat en cinc capítols.

El primer capítol es presenta la definició del projecte indicant quins són els objectius marcats en aquesta assignatura així com una planificació temporal.

El següent capítol passem a estudiar els principis bàsics dels protocols de veu IP basant-nos en l'estudi d'un en concret que és l'escollit per desenvolupar la part de veu en la nostre aplicació, el protocol SIP. En aquest capítol es presenten els aspectes fonamentals d'aquest protocol, amb l'objectiu d'introduir al lector en aquells aspectes necessaris pel desenvolupament d'una aplicació similar a la obtinguda en aquest projecte. L'enfocament d'aquest capítol és teòric.

El capítol tercer ens centrarem en l'estudi de el servidor d'extensions Asterisk des d'un punt de vista d'administrador del mateix. En un primera part introduirem al lector en la nomenclatura emprada en aquest domini, per passar en la següent part a aspectes més pràctics d'instal·lació d'un servidor Asterisk. Tot i que la potència del propi sistema és enorme, ens em centrat en aquells aspectes fonamentals necessaris pel nostre projecte.

En el quart capítol farem una breu introducció al protocol de MI Jabber, així com la posada en marxa d'un servidor d'aquest mateix sistema.

En l'últim capítol és presenta l'anàlisi i desenvolupament de l'aplicació en qüestió.

CAPÍTOL 1 – DEFINICIÓ DEL PROJECTE

1.RESUM EXECUTIU

Actualment hi ha dos sistemes de comunicació en temps real que estan aconseguint nivells d'ús molt alts: sistemes de missatgeria instantània (IM) d'una banda i dels sistemes de comunicació de veu per Internet (veuIP) d'un altra. Degut al fort lligam que existeix entre ambdós sistemes cada cop són més les empreses especialistes en comunicació que ofereixen solucions integrades . Alguns exemples d'aquest fet són els serveis que ofereix Skype, o el propi Google amb GoogleTalk. El principal problema d'aquestes solucions és que són sistemes tancats basats en tecnologies propietàries, lligant d'aquesta manera a l'usuari final.

D'altra banda existeix en el mercat un servidor d'extensions telefòniques (centraleta telefònica) anomenat Asterisk. Aquest programari a part de realitzar la gestió pròpia d'una centraleta telefònica, ofereix altres serveis com són els sistemes d'integració de tecnologies variades (gateway): veu analògica, veu IP, protocols de veu IP variats (SIP, IAX,..), i qualsevol cosa que es pugui programar.

És en aquest punt on es planteja el desenvolupament d'un projecte el qual integri els dos sistemes de comunicació esmentats (veuIP i MI) usant protocols oberts, aprofitant al mateix temps de la potència d' Asterisk. D'aquesta manera obtindrem un producte obert, de presentacions molt similars a les presents en les solucions privades, i amb una nivell molt més elevat d' escalabilitat respecte aquests darrers, esdevingut en gran mesura per la potència inherent del servidor Asterisk.

Per aconseguir el nostre objectiu es disposarà d'una sèrie de llibreries que implementin els protocols emprats. Com usarem protocols oberts podrem usar les implementacions que més ens agradi sempre i quan compleixi el principi de llibertat. Si es donés el cas de no trobar cap implementació lliure podríem arribar a desenvolupar la nostra pròpia.

D'altra banda també es farà ús de sistemes oberts com el propi servidor Asterisk, així com sistemes de MI basats en Jabber el qual ens permetrà gestionar aquest tipus de comunicació.

La durada prevista per aquest projecte és de quatre mesos.

2. DESCRIPCIÓ DEL PROJECTE

Actualment són molts els sistemes de comunicació disponibles a través de la xarxa Internet, generalment basats en dos models de comunicació: diferida i en temps real. Els primers han estat presents des dels inicis de la xarxa degut en part als pocs recursos necessaris per la seva implantació, els quals és limitaven a servidors capaços de processar gran volums d'informació en temps diferit i re enviar-la cap al destinatari. Els segons en canvi, requereixen de recursos més elevats, doncs el fet d'haver una interactivitat en temps real, produeix una càrrega més elevada tant en els servidors (processen en temps real), com de les xarxes de comunicació (missatges d'anada i tornada constants). És per aquest motiu que la implantació d'aquests sistemes no ha estat possible fins que la xarxa Internet no ha ofert amplades de banda prou elevades per suportar-les.

Un cop el tema de les comunicacions s'ha solventat han anat apareixent sistemes de comunicació en temps real, que oferien funcionalitats molt similars a altres sistemes de comunicació més tradicionals com el telèfon. Primer van ser els sistemes de missatgeria instantànea, després els sistemes de comunicació de veuIP, i actualment sistemes de videotrucades.

La gran majoria de solucions han estat implantades per empreses privades com yahoo, msn, google, skype entre d'altres, optant tots ells per solucions privades i incompatibles entre elles.

M'entres tant, en el món del programari lliure s'ha anat implementant sistemes oberts que oferisin les mateixes carecterístiques, però optant per models oberts i lliures. Tal com sol passar en aquest tipus d'entorns les eïnes que apareixen tot i ser més potents que les comercials els hi falta mecanismes de promoció, i programari fàcils d'emprar per gestionar aquests sistemes.

Davant d'aquesta realitat s'ha plantejat fer el desenvolupament d'una aplicació que integri alguns d'aquests sistemes de comunicació.

2.1. Objectius del projecte

Els objectius principals d'aquest projecte són,

1. Implantació de sistemes oberts que permetin la comunicació de VeuIP i MI.

2. Desenvolupament d'una aplicació que integri aquest dos sistemes de comunicació.

2.2 Resultats esperats

1. Obtenir una eïna de programari de codi obert, disponible per tota la comunitat, la qual ofereixi a aquesta característiques semblants a les obtingudes en solucions privades.
2. Usar el servidor Asterisk el qual ens permetrà crear xarxes de comunicació veuIP privades amb prestacions i escalabilitat molt elevades.
3. Adquirir coneixements i experiència en aquest sistemes de comunicació.
4. Adquirir coneixements i experiència en el desenvolupament d'aplicacions d'escriptori en Java.

2.2.Anàlisi de riscos

- Inexperiència en desenvolupament en Java

Tot i d'haver fet les pràctiques de la carrera en Java, l'estudiant té poca experiència en el desenvolupament d'aplicacions en aquest llenguatge, sobretot en aplicacions d'escriptori que és el cas que ens pertoca. A favor cal dir que avuí en dia Java és el llenguatge més emprat en aplicacions de codi obert (ha superat fa poc a c++ en el repositori Sourceforge) i el nombre de llibreries que podem trobar en aquest entorn és molt superior a qualsevol altre la qual cosa ens proporcionarà més marge de maniobra.

- Inexperiència en el domini del problema

Un altre factor de risc a considerar és la nul·la experiència que l'estudiant disposa en el desenvolupament d'aplicacions de comunicació per veuIP, missatgeria instàntanea així com en la configuració de centraletes telefòniques. Aquest fet pot desviar considerablement la planificació del projecte.

3. ABAST DE LA PROPOSTA

La proposta d'estudi i desenvolupament d'eines d'integració de veuIP amb Missatgeria instantània i centraletes telefòniques consta de dues etapes ben diferenciades,

- Estudi i implantació dels sistemes
  - Instal·lació d'un sistema basat en Asterisk.
  - Implantació d'un sistema de Missatgeria instànea Jabber.

- Desenvolupament d'una aplicació que integri els tres dominis subjacents
  - Avaluació de llibreries a utilitzar.
  - Desenvolupament d'aplicació.

3.1 Estudi i implantació dels sistemes

En la primera etapa d'aquest projecte és farà l'estudi i implantació dels sistemes de comunicació necessaris pel desenvolupament de l'aplicació.

3.1.1 Instal·lació d'un sistema basat en Asterisk

El primer sistema que s'implantarà, i possiblement el més complexe degut al domini del mateix és la centraleta de telèfon software Asterisk.

Aquest sistema ens permetrà emular una centraleta telefònica la qual ens proporcionarà la comunicació dels clients de veuIP que conformaran la nostra xarxa privada.

Un cop tinguem la centraleta montada haurem de configurar clients i provar la bondat de les nostres configuracions.

3.1.2 Implantació d'un sistema de Missatgeria instantània Jabber.

L'altre sistema que caldrà implantar és un servidor de missatgeria instantània que ens gestionarà aquest tipus de comunicacions.

Tal com ha passat en els sistemes de veuIP, quan tinguem instal·lat el nostre servidor de MI, haurem d'instal·lar i configurar clients d'aquest per realitzar proves.

3.2 Desenvolupament de l'aplicació

El segon gran objectiu serà crear una aplicació client que integri els sistemes de comunicació considerats amb els sistemes estudiats en l'etapa anterior.

3.2.1.Avaluació de llibreries a utilitzar.

Un primera etapa consistirà en avaluar diferents llibreries existents en el mercat que s'empraran en el desenvolupament de l'aplicació client.

3.2.2 Desenvolupament d'aplicació.

Un cop es disposi del conjunt de llibreries necessàries s'iniciarà el desenvolupament de l'aplicació client.

4. ORGANITZACIÓ DEL PROJECTE

A continuació passem a descriure les activitats desenvolupades en aquest projecte així com el temps previst per cadascuna d'elles.

4.1. Relació de tasques

4.1.1. Implantació de la centraleta Asterisk

Aquesta activitat constarà de tres fases:

- Lectura de documentació relacionada: això ens permetrà sentar les bases d'aspectes relacionats amb la gestió de centraletes telefòniques.
- Implantació d'una centraleta asterisk: aquesta part més pràctica consistirà en la posada en marxa d'un sistema basat en asterisk.
- Configuració de clients de veuIP: aquesta part ens permetra fer proves amb la centraleta instal·lada. És realitzarà paral·lelament amb la configuració pròpia de la centraleta

Durada esperada: 2 setmanes[1]

4.1.2. Anàlisi de llibreries de gestió d'asterisk

Un cop finalitzada l'activitat anterior pasarem a fer un estudi exhaustiu de les llibreries que permeten l'accés a la centraleta des d'un punt de vista de programació. Aquest estudi constarà de dues fases:

- Recerca d'informació sobre aquest tipus de llibreries
- Realització de petites aplicacions que ens permetin posar a prova les mateixes.

Duració esperada: 1 setmana

4.1.3. Anàlisi de llibreries per la comunicació de veuIP

A continuació ens dedicarem a fer l'estudi i valoració de llibreries

1 Les setmanes no són de 40 hores, sinó de 6h que són les hores corresponents de dividir 9 crèdits * 10 hora/crèdit / 16 setmanes = 90hores/16setmanes=6h/setmana.

relacionades amb el protocol de veuIP, el qual constarà també de dues etapes:

- Estudi previ d'algun protocol de veuIP: aquest estudi no serà exhaustiu ja que l'objectiu no és implementar el protocol, sino familiaritzar-se amb conceptes presents en aquest domini.

- Realització de petites aplicacions que ens permetin posar a prova les llibreries escollides.

Duració esperada: 2 setmanes

4.1.4. Implantació d'un servidor Jabber

Un cop arrivem a aquest punt tindrem solucionat la comunicació per mitjà de veu, és doncs el moment de fer el mateix però amb un sistema de Missatgeria instantànea. Distingirem dues fases,

- Instal·lació d'un servidor de MI basat en Jabber, haurem doncs d'avaluar entre diferents opcions escollint la que més s'ha adecui a les nostres necessitats.
- Instal·lació de clients de MI els qual ens permetran provar el nostre servidor.

Duració esperada: 2 setmanes

4.1.5. Anàlisi de llibreries per la comunicació via Jabber

El següent pas serà avaluar les llibreries que farem servir per la comunicació entre els clients i el servidor Jabber. Per fer-ho dividirem l'activitat en dues fases,

- Lectura de documentació relacionada amb el protocol Jabber, la qual ens permetrà senar les bases a nivell de programador d'aquest model de comunicació.
- Realització d'aplicacions petites que ens permetin provar aquestes llibreries.

Durada esperada: 2 setmanes

4.1.6. Disseny i desenvolupament de l'aplicació client

Un cop hagem arribat en aquest punt obtindrem ja estarem en condicions de desenvolupar una aplicació que integri tot els aspectes realitzats en les etapes anteriors. Distingirem les següents etapes,

- Disseny de la interfície, on definirem l'aspecte visible de la nostra aplicació.
- Disseny de l'aplicació, on definirem l'estructura interna de la nostra aplicació: quantes classes programarem, com ho farem, quins patrons aplicarem.
- Desenvolupament de l'aplicació: programació pròpiament dita.
- Proves que permetin valorar la bondat de la nostra aplicació.

Durada esperada: 4 setmanes

4.1.7. Desenvolupament de la memòria

Aquesta activitat constarà de la redacció pròpiament dita d'aquesta memòria.

Durada esperada: dues setmanes repartides al llarg del semestre

4.2.Fites principals

Le fites principals definides són

- Inici del projecte: aquesta és la fase inicial present en tot projecte.
- Sistema de comunicacions per veuIP emprant Asterisk: aquesta fita s'assolirà quan els aspectes relacionats amb la comunicació per veu arribin a la seva fi, el quals inclouen també els relacionats amb la gestió del servidor Asterisk.
- Sistema de comunicacions per MI emprant Jabber: aquesta fita s'assolirà quan els aspectes relacionats amb la comunicació per MI emprant Jabber arribin a la seva fi.
- Desenvolupament de l'aplicació: un cop hagem integrat els dos sistemes de comunicació en un aplicació client haurem assolit aquesta fita.
- Lliurament final: finalment la darrera fita serà l'entrega del projecte.

CAPÍTOL 2 -PROCOLS DE VeuIP

En l'actualitat la telefonia a través d'Internet s'està popularitzant gràcies a la proliferació d'empreses que ofereixen aquest servei a tot tipus d'usuaris amb un cost menor que la telefonia tradicional. Cada cop són més les empreses que ofereixen aquesta via de comunicació, d'entre les quals podem destacar Microsoft i Skype pel seu major impacte. La primera va implantar ràpidament la comunicació de veuIP dins del seu client de MI Messenger, el qual permetia a aquest la comunicació per veu gratuïta a tots els usuaris d'aquesta xarxa. La segona ha anat més enllà oferint la commutació a altres sistemes més tradicionals com el telèfon convencional o el mòbil per un preu inferior al de la trucada convencional (tan sols pagues el tram que no és Internet, en aquest cas Xarxa pública commutada o GSM).

En aquest capítol veurem els fonaments d'aquestes aplicacions, així com del nostre projecte, fent una breu introducció als protocols de comunicació de veuIP. Per fer-ho estudiarem el protocol SIP emprat en el nostre projecte, la justificació del qual esdevé pels següents punts:

- Àmpliament provat.
- Nombre elevat de llibreries que l'implementen.
- Gran volum d'informació sobre el protocol.
- Amb un passat, present i futur reconegut per organismes de reconegut prestigi.
- Amb un gran nombre de clients (soft i hard) que l'implementen.

1 PROTOCOL DE SENYALITZACIÓ SIP

Funcions de SIP

Les funcions principals de SIP són:

- SIP permet la localització d'usuaris, usant unes taules de traducció de nom d'usuari a adreça actual en la xarxa.
- SIP permet negociació de la sessió, oferint als participants un mecanisme perquè puguin decidir sobre les característiques d'aquesta en funció de la implementació particular de cadascun.
- SIP permet gestió de la trucades, com per exemple, realitzar noves trucades, finalitzar-les, transferir-les entre d'altres.
- SIP permet canviar les característiques de la sessió quan aquesta ja està iniciada.

Components de SIP

Missatges, transaccions, diàlegs i trucades

La unitat de transferència entre clients i servidors són els missatges formats per texts.

Les transaccions són entre clients i usuaris i comprometen tots els missatges que hi ha entre una petició i una resposta.

Diàlegs són relacions d'igual a igual entre dos Agents d'usuari les quals persisteixen durant un temps.

Una trucada és el conjunt de diàlegs involucrats.

Agesnts d'usuari

Les entitats que interactuen en SIP s'anomenen agents d'usuari (User Agents), de les quals hi ha de dos tipus:

- Agents d'usuari Client (UAC): generen peticions i les envien als servidors.
- Agents d'usuari Servidor (UAS): reben peticions, les processen i generen respostes.

Clients

Els clients poden ser programes (softphone) o dispositius físics (hardphones), els quals generen peticions quan intenten comunicar-se amb un altre client.

Servidors

Els servidors són els encarregats de processar les peticions d'usuari i fer possible la comunicació entre els diferents clients. Podem diferenciar quatre tipus de servidors:

- **Servidors Regisrar:** És el servidor on es registren els clients per estar visibles a la xarxa.
- **Location server:** Aquest tipus de servidors emmagatzemen les adreces dels clients registrats .
- **Servidors proxies (proxy server):** Fan d'intermediari entre clients, i permeten una major escalabilitat al sistema, així com una configuració molt més senzilla en els clients.
- **Servidors de redirecció (redirect server):** són els encarregats de redireccionar peticions cap una adreça diferent indicant al sol·licitant que ha d'anar per una ruta diferent a l'habitual. Això passa quan el receptor s'ha mogut de la seva adreça normal temporal o permanentment.

Esquemes de comunicació

Proxies

En la següent figura podem veure un esquema de comunicació emprant servidors proxies a la banda del client que rep la trucada. El diàleg s'inicia quan un usuari qualsevol vol establir comunicació amb un usuari del domini xarxa.cat (usr1@xarxa.cat). Com no coneix la direcció exacta de usr1@xarxa.cat, fa ús d'un servidor proxie, el qual té assignat una IP

pública coneguda pel client. Aquest per mitjà d'el servidor de localització obté l'adreça exacta de l'usuari us1@xarxa.cat, en aquest cas és una adreça IP privada. Un cop localitzat el proxie li envia la comanda INVITE a aquest, i aquest li respon acceptant (OK) a aquest, que alhora reenvia la resposta cap al client originador de la trucada. Ara el client ja coneix la localització exacta del usr1@xarxa.cat[2] i a partir d'aquest moment li enviarà totes les comandes directament a ell sense passar pel proxi. Així ho fa en la darrera comanda ACK que finalitza la transacció d'inici de trucada.

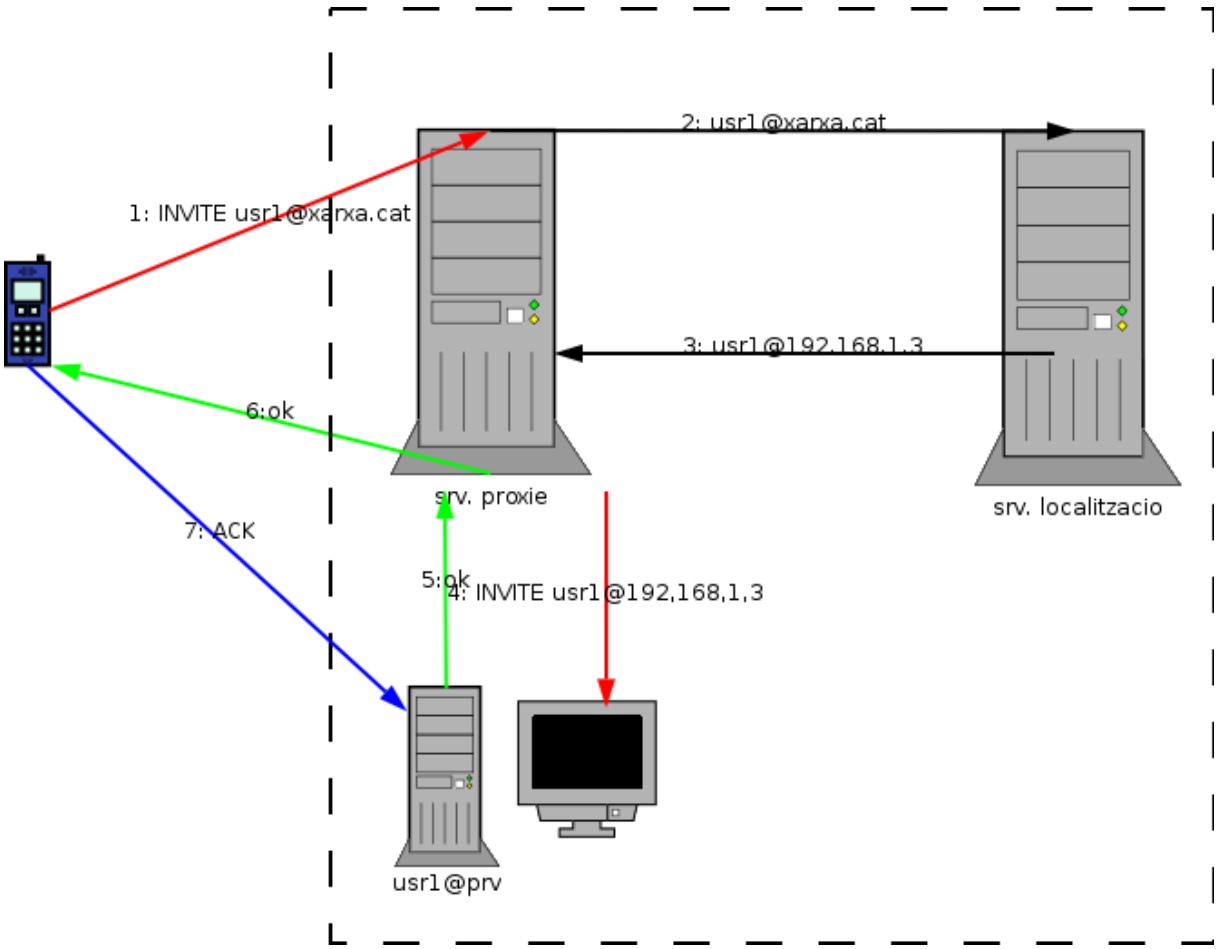
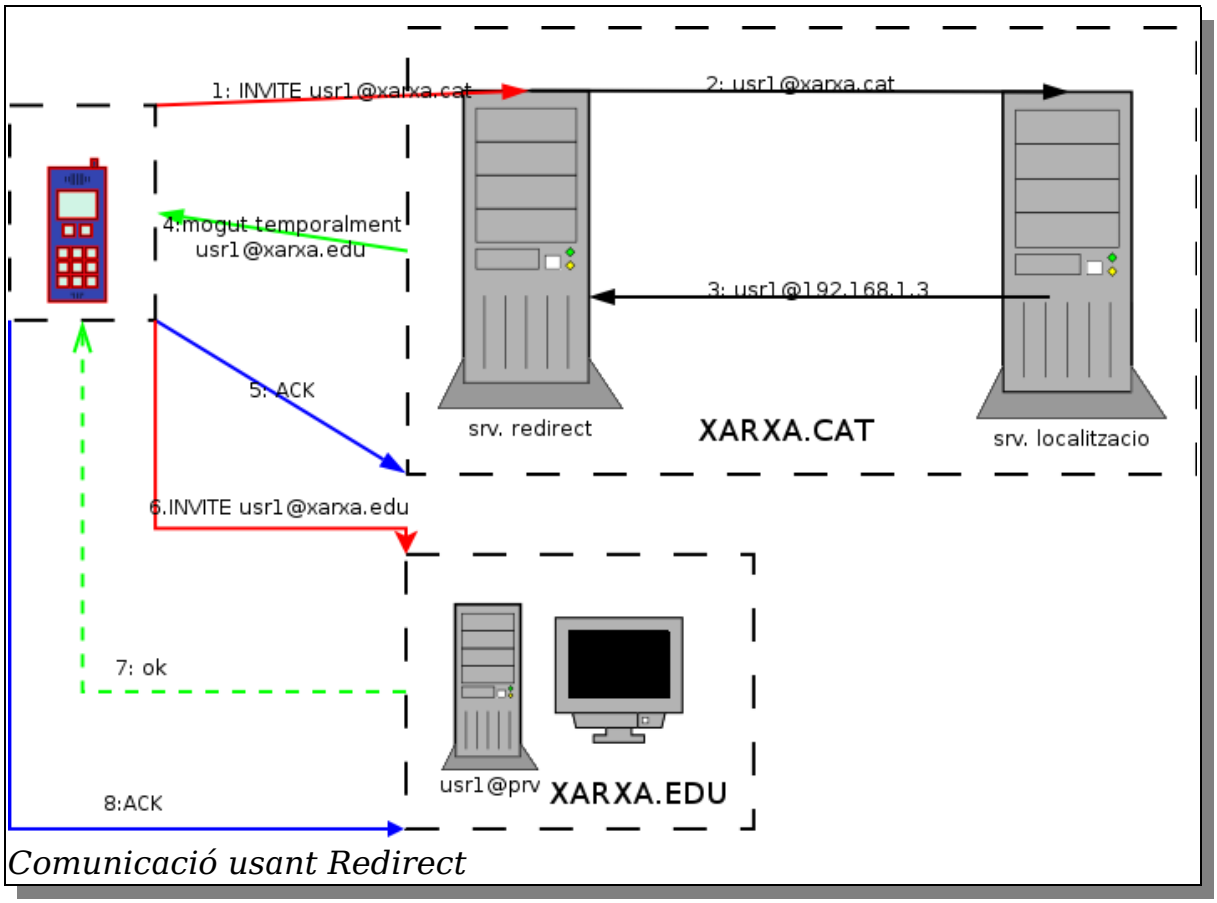


Illustration 1: Comunicació usant proxie

2 l'@IP privada s'haurà de mapejar en alguna pública per mitjà de NAT, però això es cosa del router de sortida

Redirect

A continuació veiem el mateix diàleg però fent ús d'un servidor redirect, el qual indica al client que s'ha mogut a una adreça que no és l'habitual usr1@xarxa.edu. Un cop el client obté aquesta informació torna a iniciar tot el procés a aquesta adreça.



Comunicació usant Redirect

2 PROTOCOL DE TRANSMISIÓ EN TEMPS-REAL (RTP)

El protocol de transmissió en temps real (RTP) és el protocol emprat per transmetre la veu en les comunicacions de veuIP que fan servir SIP. Aquest protocol ofereix serveis d'entrega punt a punt per dades que requereixen característiques pròpies del temps real, com per exemple les dades d'àudio o vídeo.

RTP és un estàndard especificat en els RFC 1889, 3550 i 3551.

Les aplicacions de temps real són aquelles que consideren el temps com l'element prioritari en la resolució del problema. Per aconseguir-ho RTP inclou mecanismes de sincronització de fluxos per mitjà de propietats temporals tal com veurem en els següents apartats.

RTP és un protocol que està dividit en dos parts:

- RTP : l'encarregat de gestionar l'entrega de les dades.
- RTPC(RTP control): l'encarregat de monitoritzar la qualitat del servei i transmetre informació sobre els participants.

Fonts de contingut i de sincronització

Una font de contingut són les dades originals enviades en paquets RTP, per exemple les dades d'àudio d'una aplicació. Diferents fonts poden ser combinades en un mateix paquet RTP per un pont (en el següent apartat), donant lloc a una font de sincronització nova.

Les fonts de sincronització identifiquen un sol contingut o la combinació de molts, per exemple, un canal d'àudio o la combinació de dos (pe. en sistemes DUAL). Aquestes estan associades a un nombre de seqüència diferencial, i poden canviar el format de les dades durant el temps (pe. la codificació).

Elements estructurals

La entrega de paquets RTP es fa a traves d'una xarxa de comunicació, dins la qual podem trobar tres tipus d'elements involucrats en aquest protocol: Sistemes terminals, Ponts (RTP-Bridges) i Traductors

(Translators).

Sistemes terminals

Són els generadors i consumidors de paquets RTP. Un sistema terminal pot actuar com un o més fonts de sincronització.

Ponts (bridges)

Reben paquets RTP de diferents fonts, els combinen d'alguna manera i reenvien el nou paquet. Poden canviar el format de les dades. Com solen rebre informació de diferents fonts de sincronització , generalment no solen estar sincronitzats, es per això que aquest elements han de fer ajustaments de temps i generar el seu propi espai temporal per l'stream combinat. El nou paquet generat porta en el camp SSRC l'identificador del Bridge, i en la CSRC els identificadors de les fonts de sincronització d'entrada.

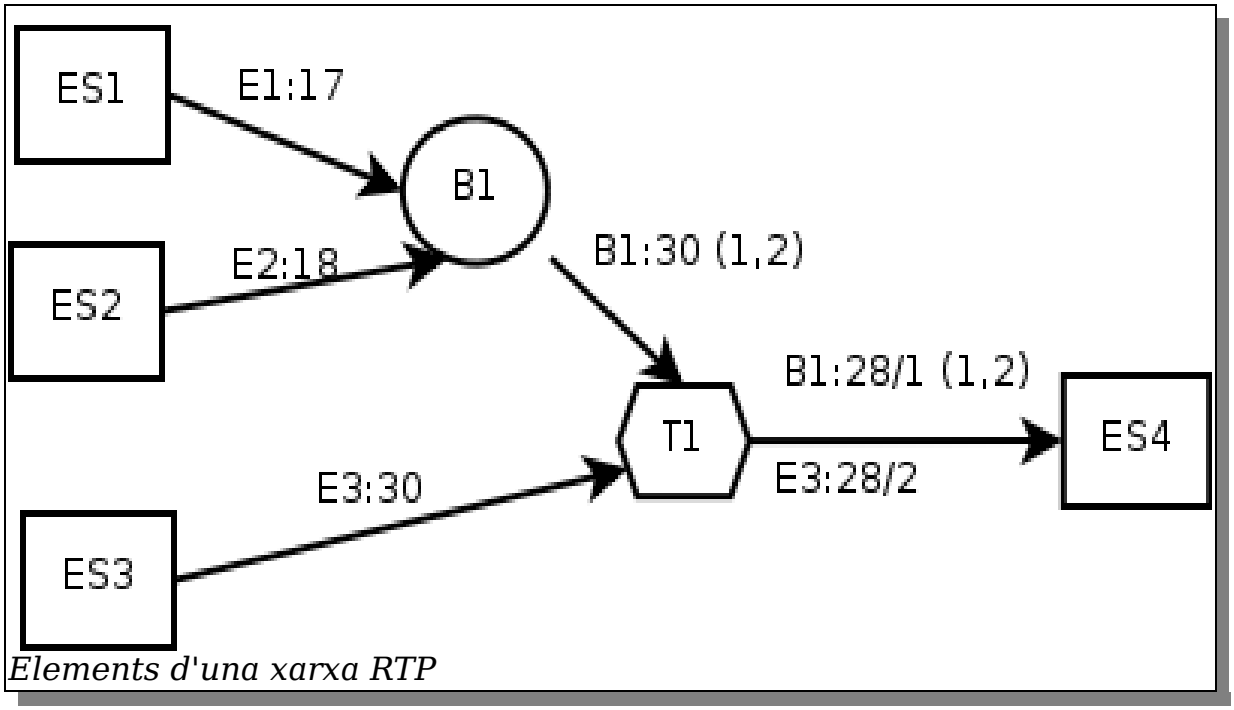
Una de les aplicacions més comuns d'aquest elements és la de canviar l'amplada de banda en enllaços de poca capacitat. Per exemple si tenim un enllaç de baixa amplada de banda, sinó disposéssim de bridges tots els elements involucrats haurien d'enviar informació amb aquesta amplada desaprofitant així la capacitat dels altres canals. Amb un bridge al costat de l'enllaç de baixa amplada de banda solucionem el problema, ja que la resta d'elements poden transmetre aprofitant tota l'amplada de banda del canal deixant pel bridge la fusió quan hagi de reenvia cap a l'enllaç de poca capacitat.

Traductors (Translators)

reenvien paquets però no canvien el nombre de seqüència ni el timestamp (tal com fan els bridges). Alguns exemples poden ser: conversió de codificacions sense barreja, conversió de multicast a unicast i filtres a firewalls.

El següent esquema representa els elements comentats. Els Esn són els sistemes finals que produeixen fonts de contingut. Els Bn són els ponts, tal com podem veure la figura hi ha un el qual barreja el contingut de

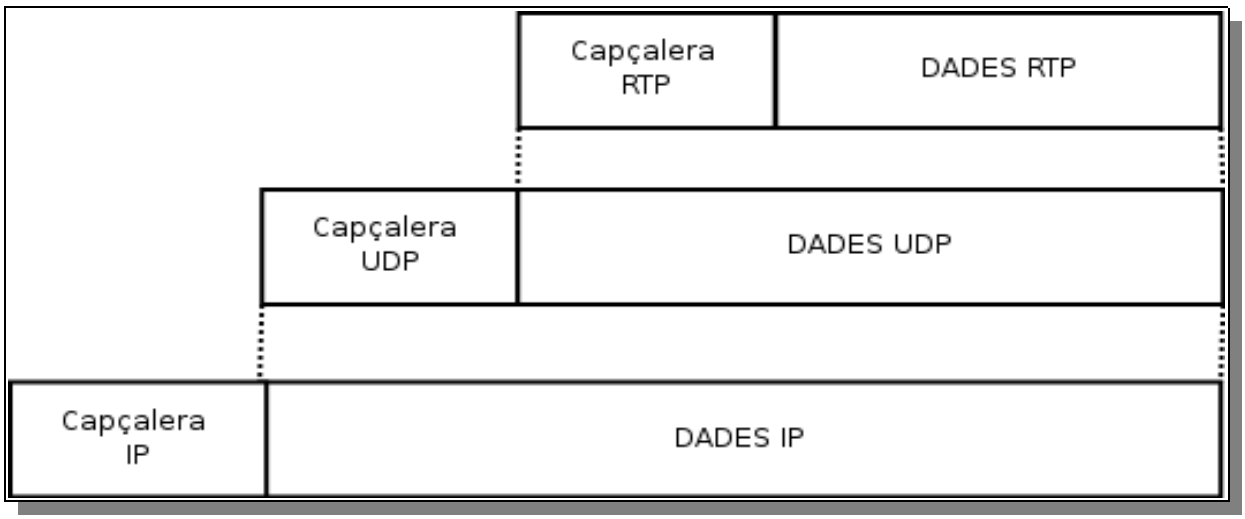
ES1 i ES2 en un sol stream, el qual durà un nou identificador de sincronització (SSRC=30) i a la llista de contribucions i posarà els elements entrants (CSRC={1,2}). L'element T1 és un traductor, el qual rep dos fonts i les retransmet (no barreja) canviant-ne només la codificació.



Elements d'una xarxa RTP

Estructura de paquets

RTP empra la seva pròpia estructura de paquets per transmetre la informació. Aquesta informació només aporta aspectes de temps real, però en cap cas RTP no defineix un protocol de transmissió, és per això que s'encapsula dins les dades del protocol UDP per fer arribar la informació d'un costat a l'altre de la comunicació.



Capçalera

CAMP	*BITS*	*DESCRIPCIÓ*
V (Version)	0-1	Nombre de versió.
P (Padding)	2-2	Si està a 1 indica que les dades incorporen bytes amb informació de padding, i l'últim byte de les dades indica quants octets hi ha.
X (Xtension)	3-3	Si està a 1 indica que a la capçalera li segueix una capçalera d'extensió.
CC (Counter CSRC)	4-7	Indica el nombre de CSRC.
M (Marker)	8	Aquest camp el poden fer servir les aplicacions pel seu propi ús.
PT (Payload type)	9-15	Indica el format de les dades RTP (la codificació).
Nombre seqüència	16-21	S'incrementa en cada paquet RTP i pot ser usat per l'aplicació per detectar paquets perduts.
timestamp	0-21	El timestamp reflecteix l'instant de mostra del primer byte de les dades RTP. Aquest ha de ser un valor d'un rellotge que s'incrementa constantment i linealment, i permet la sincronització
SSRC identifier	0-21	És l'identificador d'una font de sincronització (un canal d'àudio, un canal de vídeo,...). És un nombre aleatori el qual no s'hauria de repetir per dos fonts diferents.
CSRC identifiers	0-21	Es la llista de les fonts de sincronització que han contribuït

Sincronització

El receptor far servir tres camps per la sincronització: SSRC, nombre de seqüència i timestamp.

Synchronization Source (SSRC)

El receptor pot rebre dades de diferents fonts, per poder tractar-les correctament necessita identificar els paquets de les diferents fonts. Això és possible pel camp SSRC.

Nombre de seqüència

El nombre de seqüència ens permet establir l'ordre dels paquets. El nombre de seqüència s'incrementa en un per cada paquet enviat, i pot ser emprat pel receptor per detectar paquets perduts. Les pèrdues o entregues desordenades pot donar-se quan hi ha problemes a la xarxa.

Timestamp

El nombre de seqüència no es suficient per ordenar temporalment els paquets de l'stream. Això és degut a que el moment de la mostra no sempre és el mateix ordre que el d'enviament, com per exemple passa amb el sistema MPEG que interpola els frames del vídeo També pot passar que paquets consecutius RTP portin el mateix timestamp, si aquests han estat generats a la vegada, per exemple paquets del mateix frame de vídeo

RTPC

Aquest protocol està basat en la retransmissió periòdica de paquets a tots els participants de la sessió amb el mateix mecanisme de distribució que les dades. Les funcions d'aquest són:

- Provenir informació relacionada amb la qualitat de les dades.
- Provenir d'informació relacionada amb el nom identificable de les fonts RTP anomenada canonical name or CNAME. Notar que el SSRC pot canviar durant el temps en canvi el CNAME no. També pot

incorporar informació adicional com l'email.

- Calcular la freqüència que els paquets s'han d'enviar. Això s'aconsegueix observant els diferents paquets de control provinents dels diferents participants.

Tipus de paquets

- SR (Sender Report): per la transmissió i recepció d'estadístiques dels participants que envien.
- RR (receiver Report): per la transmissió i recepció d'estadístiques dels participants que no envien.
- SDES: Elements de descripció com el CNAME.
- BYE: indica finalització de la participació.
- APP: Funcions especifiques de l'aplicació.

3 PROTOCOL DE DESCRIPCIÓ DE SESSIÓ (SDP)

Aquest protocol és usat per descriure sessions multimèdia en un format que comprensible per tots els participants de la xarxa. En funció d'aquesta descripció es pot decidir qui, com o quan els participants poden afegir-se a un conferència.

El propietari de la conferencia envia missatges multicast a la xarxa que contenen descripció de la sessió, com per exemple, el nom del propietari, el nom de la sessió, la codificació, el temps,... En funció d'aquesta informació els receptors poden prendre decisions sobre la participació a la sessió. Generalment SDP és encapsulat al cos d'un missatge SIP.

Descripció de la sessió

La informació que proporciona aquest protocol és:

- Nom de la sessió i propòsit.
- Moment en el temps que la sessió és activa.
- Els mitjans que composen la sessió.
- Informació sobre l'amplada de banda necessària per la conferència.
- Informació de contacte sobre la persona responsable de la sessió.
- Tipus de mitjà: vídeo, àudio, ...
- Protocol de transport: RTP/UDP/IP, H.320,..
- Format del mitjà: Mpeg vídeo, AVI vídeo,..
- Si es tracta d'una sessió multicast: l'adreça i el port de transport.
- Si es tracta d'una sessió unicast: l'adreça remota i port.

Format de paquets i paràmetres

Els paquets SDP són completament textuals usant el joc de caràcters ISO 10646 amb codificació UTF-8. Ha estat dissenyat en format textual en contraposició al binari per temes de portabilitat, possibilitant gran varietat de transports. Un altre característica d'aquest protocol és que ha estat dissenyat per ser extremadament compacte i així ocupar poc ample de banda. Per altra banda, els missatges poden ser transportats per mitjans poc fiables o poden ser malmesos per servidors intermediaris, és per això que la codificació s'ha dissenyat seguint un ordre i unes normes

de format, per així detectar molts errors i descartar-los.

Una sessió SDP consisteix en un nombre de línies de text de la forma <codi>=<valor>, sent <codi> un caràcter que identifica la comanda, i <valor> és un text estructurat el contingut del qual depèn del codi.

Codis

Codi	***Descripció***	***exemple***
v	versió del protocol	0
o	propietari/creador i identificador de sessió	cmiralles 2890844526 2890842807 IN IP4 80.72.64.4
s	nom de la sessió	SDP seminar
i*	informació de la sessió	Un seminari sobre SDP
u*	uri de la descripció	http://www.uoc.edu/cmirallesp/SIP/rfc.pdf
e*	adreça email	cmirallesp@uoc.edu
p*	nombre de telefon	
c	informació de connexió	IN IP4 224.2.17.12/127
b*	informació d'ample de banda	
k*	clau d'encriptació	
a*	0 o més línies d'atributs de sessió	recvonly orient:portrait
t	temps que la sessió és activa	2873397496 2873404696
r*	repetir 0 més vegades	
m	nom del mitjà i adreça de transport	àudio 49170 RTP/AVP 0 vídeo 51372 RTP/AVP 31

CAPÍTOL 3 - ASTERISK

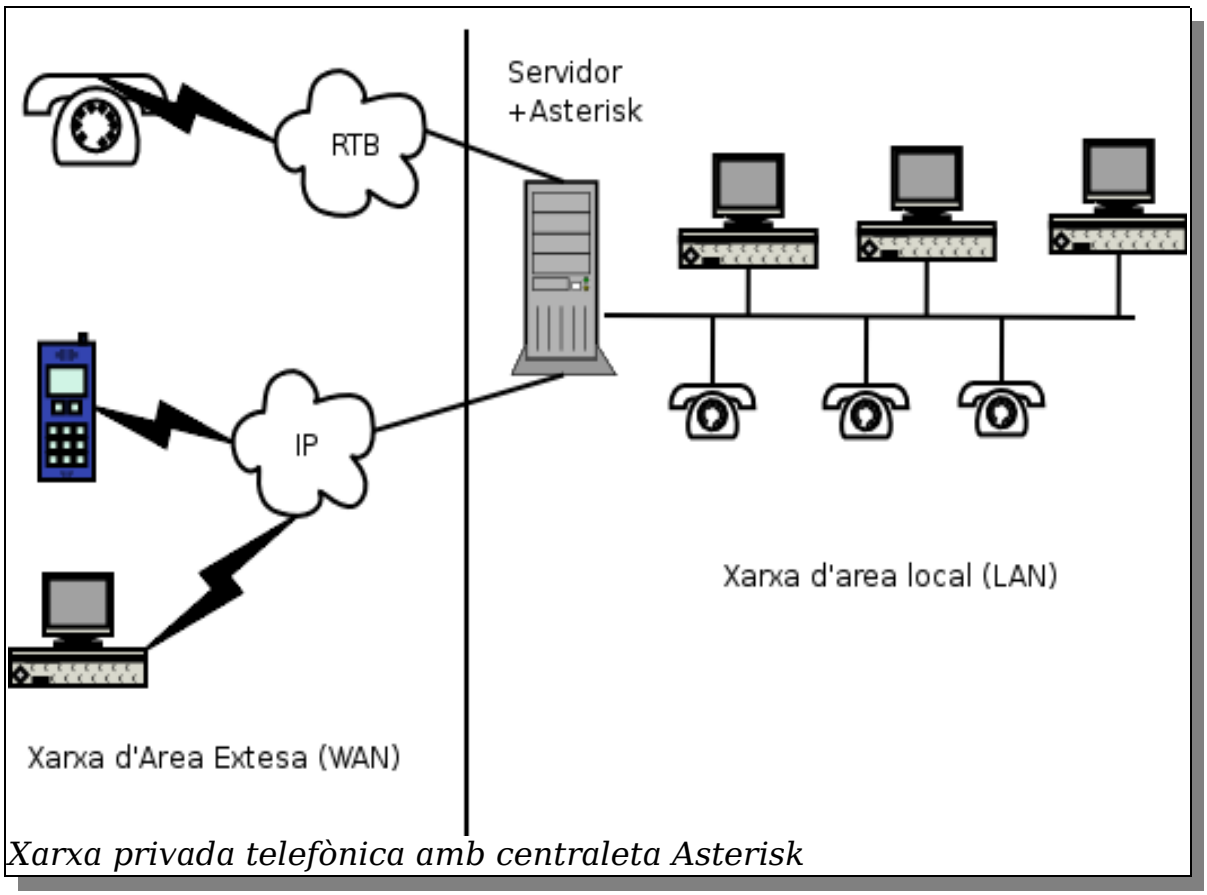
En aquest capítol ens centrarem en l'estudi de la centraleta software Asterisk. Els tres primers punts ens centrarem en els conceptes fonamentals d'aquest servidor. En el quart punt ens centrarem en un aspecte anomenat Asterisk RealTime i que és necessari per poder modificar dinàmicament l'entorn del servidor, i que degut a la complexitat del mateix s'ha tractat en un punt a part. En el darrer punt tractarem dels models de programació disponibles.

1 Introducció

Asterisk és un software que emula les característiques d'una centraleta telefònica tradicional i n'amplia les funcionalitats. Les característiques principals d'aquest software són:

- Integració de sistemes de veuIP (gateway): Aquesta característica permet la comunicació entre clients de veuIP que emprin protocols diferents: SIP, IAX, H.323.
- Implementa una xarxa privada de telefonia (PBX): aquesta característica permet crear una xarxa privada de comunicació entre els clients de la mateixa.
- Servidor de resposta de veu personalitzable: ens permet crear respostes interactives automatitzades.
- Servidor de conferències: permet crear conferències.
- Cues de trucades i trucades predictives.
- Connexió amb la xarxa pública commutada.

En la següent figura podem veure un esquema genèric on dispositius de diferents xarxes és comuniquen entre ells per mitjà d'una centraleta Asterisk



Xarxa privada telefònica amb centraleta Asterisk

2 Canals (channel)

En terminologia Asterisk un canal defineix un enllaç entre un terminal i la centraleta Asterisk Generalment es distingeixen tres tipus de canals,

- Un canal (o més) per connectar el nostre servidor a la xarxa telefònica clàssica, i que permet rebre/fer trucades de/cap l'exterior.
- Diferents terminals que fan ús de la xarxa clàssica (telèfons, fax,..).
- Diferents terminals que fan ús d'una xarxa IP (softphone, hardphone..).

Els canals es defineixen a diferents fitxers de configuració depenent del tipus de canal que sigui, alguns exemples són:

```
/etc/Asterisk/zaptelf.conf  Per connexions a linees FXO i FXS (rj45).
/etc/Asterisk/sip.conf      Per connexions per mitjà del protocol SIP.
/etc/Asterisk/aix.conf      Per conexions per mitjà del protocol AIX.
```

Canals podem trobar de tres tipus, en funció de la direcció de les trucades,

- Canals d'entrada *(user)*: només poden rebre trucades.

- Canals de sortida *(peer)*: només poden fer trucades.
- Canals d'entrada/sortida *(friend)*: poden fer i rebre trucades.

3 Pla de trucades (dialplan)

El pla de trucades defineix les extensions actives de la nostre xarxa privada, així com les accions a realitzar quan un usuari prem una d'aquestes des del seu terminal.

Tot el que es pot fer amb Asterisk es defineix en aquest pla, per això és una de les parts més complexes del servidor, i l'estudi del mateix esdevé una necessitat. Tanmateix, nosaltres només ens centrarem en aquells trets més útils per el nostre projecte.

En Asterisk tot el pla de trucades es descriu en un fitxer anomenat */etc/Asterisk/extensions.conf*

Contexts (context)

És una abstracció que permet dividir el pla de trucades en seccions cadascuna de les quals té associat un grup d'extensions. Així doncs, els contexts ens permeten aïllar les extensions de diferents seccions, oferint una estructura de segmentació lògica.

Els contexts es defineixen amb un nom entre cotxet, pe. [administració].

Extensions

Les extensions són els identificador que ha de marcar un terminal per connectar-se amb un altre i en Asterisk es defineixen amb el patró,

exten => nom,prioritat,acció()

```
Alguns exemples,
exten => 123,1,Answer()
exten => 123,2,HangUp()
```

Hi ha una extensió especial anomenada “s” d'start, que s'aplica quan entrem a la centraleta des de fora de la xarxa privada.

Prioritats

Les prioritats formen part de la definició de l'extensió, i indiquen quina

seqüència d'accions ha seguir Asterisk per processar una extensió. Les prioritats han d'anar numerades consecutivament sense saltar-se cap nombre. Per facilitar modificacions i evitar errors en el nostre pla de trucades, existeix la prioritat *n*, que suma una unitat a la prioritat anterior (així només cal numerar la primera).

Accions

Les accions indiquen a Asterisk quin procediment ha d'executar quan està processant una extensió. A continuació veiem algunes interesants i útils aplicacions,

Answer()	Respon a la trucada, i realitza la configuració de la mateixa.
Hangup()	Finalitza la trucada.
Playback(filename)	Executa el fitxer de so "filename" (no posar extensió).
Background(filename)	Executa el fitxer de so "filename" però quan rep una pulsació passa el control a l'extensió que té el mateix nombre que ha pres. Típicament aquesta acció és usada per crear menús de veu. Si rep un entrada invàlida salta a l'extensió "*i*" (s'ha de considerar).
goto(context,extension,priority)	Salt incondicional a l'extensió.
Dial()	Realitza una trucada.
Meetme()	Per crear sales de conferencia.

Cal dir que d'accions en Asterisk n'hi ha moltes, i ofereixen a l'administrador del sistema un potència extrema, l'estudi de les quals queda fora de l'abast del nostre projecte.

Exemple

A continuació presentem un petit pla de trucades on es presenten la majoria dels aspectes comentats en aquesta secció.

```
[incoming]
; Aquest context permet connectar les trucades exteriors amb les diferents extensions.
exten => s,1,Answer( ) ;despenjar
```

```
exten => s,2,Background(enter-ext-of-person) ; seleccioni una extensió (101/102)?

;Pla per l'extensió 101
exten => 101,1,Dial(Zap/1,10) ; trucar al canal Zap/1
exten => 101,2,Playback(vm-nobodyavail) ;si no agafen la trucada, missatge "no disponible"
exten => 101,3,Hangup( ) ;penjar
exten => 101,102,Playback(tt-allbusy) ;si comunica, missatge "ocupat".
;la prioritat 102 ve de n+101 sent n la prioritat on s'ha executat Dial (en aquest cas 1)
exten => 101,103,Hangup( )

;Pla per l'extensió 102
exten => 102,1,Dial(SIP/Jane,10)
exten => 102,2,Playback(vm-nobodyavail)
exten => 102,3,Hangup( )
exten => 102,102,Playback(tt-allbusy)
exten => 102,103,Hangup( )
exten => t,1,Playback(vm-goodbye)
exten => t,2,Hangup( )

[internal]
;Aquest context connecta dos canals entre ells internament
exten => 101,1,Dial(Zap/1,,r)
exten => john,1,Dial(Zap/1,,r) ;per nom
exten => 102,1,Dial(SIP/jane,,r)
exten => jane,1,Dial(SIP/jane,,r)
```

4 Configuració per mitjà de BBDD (RealTime)

Asterisk ofereix la possibilitat d'emmagatzemar les dades dels fitxers de configuració en Bases de Dades, facilitant així la gestió de la configuració a programes externs, o dit d'una altra manera, aquest sistema ens permet gestionar dinàmicament l'entorn d'aquest sense tenir que reiniciar el servidor cada cop que fem un canvi. En la literatura relacionada s'hi refereix a aquesta característica com a RealTime per la manera en que s'actualitzen les dades.

Asterisk permet dos tipus de gestions per mitjà de BBDD:

- Estàtica (static): Les dades es carreguen un sol cop, quan iniciem Asterisk.
  - ✓ L'avantatge d'aquest mode és que la càrrega del servidor de BBDD és baix, i Asterisk pot treballar encara que la BD caigui.
  - ✓ L'inconvenient és que això no és temps real, i Asterisk s'ha de tornar a carregar cada cop que modifiquem la BD.
- Dinàmica (dynamic): Les dades es carreguen quan s'inicia una trucada i es descarreguen quan aquesta finalitza.

- ✓ L'avantatge d'aquest model és que Asterisk sempre té les dades actualitzades, i no cal reiniciar-lo cada cop que modifiquem la BD.
- ✓ L'inconvenient per contra és que la disponibilitat d'Asterisk depèn directament de la del Servidor de BD, per tant si aquest cau, tot el sistema cau.

D'altra banda Asterisk ofereix dos tipus d'accés a Bases de Dades:

- Mètode MySql: Treballa amb SGBD MySql.
- Mètode ODBC: Treballa amb qualsevol SGBD del que disposem Driver ODBC.

Mapeig fitxers a taules

El fitxer que mapeja els fitxers de configuració a taules és diu */etc/asterisk/extconfig.conf* i té estructura diferent en funció de si fem Static o Dynamic Real time.

- Static Real Time (*/etc/asterisk/extconfig.conf)*

```
conf filename    =>  driver,   databasename   [,table_name]
queues.conf      =>  odbc,     asterisk,      ast_config
sip.conf         =>  mysql,    asterisk
iax.conf         =>  ldap,     MyBaseDN,      iax
```

El primer cas mapegem el fitxer de cues a la taula ast_config de la BD asterisk via ODBC. El segon cas mapegem el fitxer de canals SIP a la taula sip.conf a la BD asterisk via MySql. El darrer cas mapegem el fitxer de canals IAX a la taula iax a la BD MyBaseDN via LDAP.

- Dynamic RealTime (*/etc/asterisk/extconfig.conf)*

```
family name     => driver,  database name   [,table_name]
sippeers        => mysql,   asterisk,       sip_peers ;canals SIP per fer
trucades
sipusers        => mysql,   asterisk,       sip_users ;canals SIP per rebre
trucades
queues          => mysql,   asterisk,       queue_table
queue_members   => mysql,   asterisk,       queue_member_table
voicemail       => mysql,   test
```

Nota: podem mapejar sippeers i sipusers a la mateixa taula.

5 Models de programació

En asterisk és disposa de dos models de programació que permeten ampliar les funcionalitats de la centraleta: AGI i Manager.

La primera s'inspira en el model dels CGIs dels servidors web, el qual consisteix en desenvolupar aplicacions, seguint una estructura determinada les quals poden ser cridades directament des del propi asterisk (des del pla de trucades).

El segon model, és l'emprat en la nostra aplicació, i consisteix en una API que publica Asterisk al mon i la qual consisteix

- Accions: com el seu nom indica, són accions que es poden solicitar desde l'aplicació client a asterisk, com per exemple: truca aquesta extensió.
- Events: són events que envia asterisk a les aplicacions que cumpleixen una interficie de programació determinada i que expresen canvis d'estat en la centraleta, com per exemple: l'usuari “Anna” s'ha registrat.

En el desenvolupament de l'aplicació s'usarà aquest d'arrer, i per activar-lo cal modificar el fitxer de configuració **/etc/asterisk/manarger.conf**

```
[general]
enabled = yes    ;activa el manager
port = 5038
bindaddr = 0.0.0.0

[manager]
secret = pa55w0rd       ;password usat per connectar-se al manager
;deny=0.0.0.0/0.0.0.0
permit=0.0.0.0/0.0.0.0 ;permet connectar-se des de qualsevol ip
;permit=209.16.236.73/255.255.255.0
;
; Authorization for various classes
read = system,call,log,verbose,command,agent,user
write = system,call,log,verbose,command,agent,user
```

6 Annex: Configuració d'ASTERISK en Dynamic RealTime

1) UnixODBC configuració

Cal modificar els fitxers /etc/odbc.ini i /etc/odbcinst.ini

;/etc/odbc.ini

```
[ODBC Data Sources]
asterisk = PostgreSQL ODBC driver
[asterisk]
Description         = PostgreSQL asterisk
Driver              = /usr/lib/odbc/psqlodbc.so
;Trace              = Yes
Trace               = No
;TraceFile          = /tmp/extodbc.log
TraceFile           = stderr
;Debugging          = Yes
;DebugFile          = /tmp/odbcdebug.log
Database            = asterisk
;Servername         = localhost
Servername          = 127.0.0.1
UserName            = asterisk
Password            = asterisk
Port                = 5432
Protocol            = 6.4
ReadOnly            = No
RowVersioning       = No
ShowSystemTables    = No
ShowOidColumn       = No
FakeOidIndex        = No
ConnSettings        =
```

;/etc/odbcinst.ini

```
[PostgreSQL]
Description         = PostgreSQL ODBC driver
Driver              = /usr/lib/odbc/psqlodbc.so
Setup               = /usr/lib/odbc/libodbcpsqlS.so
Debug               = 1
CommLog             = 1
UsageCount          = 1
```

2) Creació de taules

```
CREATE TABLE extensions_conf (
id serial NOT NULL,
context character varying(20) DEFAULT '' NOT NULL,
exten character varying(20) DEFAULT '' NOT NULL,
priority smallint DEFAULT 0 NOT NULL,
app character varying(20) DEFAULT '' NOT NULL,
appdata character varying(128)
);
```

```
CREATE TABLE cdr (
calldate timestamp with time zone DEFAULT now() NOT NULL,
clid character varying(80) DEFAULT '' NOT NULL,
src character varying(80) DEFAULT '' NOT NULL,
dst character varying(80) DEFAULT '' NOT NULL,
dcontext character varying(80) DEFAULT '' NOT NULL,
channel character varying(80) DEFAULT '' NOT NULL,
dstchannel character varying(80) DEFAULT '' NOT NULL,
lastapp character varying(80) DEFAULT '' NOT NULL,
lastdata character varying(80) DEFAULT '' NOT NULL,
duration bigint DEFAULT 0::bigint NOT NULL,
billsec bigint DEFAULT 0::bigint NOT NULL,
disposition character varying(45) DEFAULT '' NOT NULL,
amaflags bigint DEFAULT 0::bigint NOT NULL,
accountcode character varying(20) DEFAULT '' NOT NULL,
uniqueid character varying(32) DEFAULT '' NOT NULL,
```

```
userfield character varying(255) DEFAULT '' NOT NULL
);

CREATE TABLE sip_conf (
id serial NOT NULL,
name character varying(80) DEFAULT '' NOT NULL,
accountcode character varying(20),
amaflags character varying(7),
callgroup character varying(10),
callerid character varying(80),
canreinvite character varying(3) DEFAULT 'yes',
context character varying(80),
defaultip character varying(15),
dtmfmode character varying(7),
fromuser character varying(80),
fromdomain character varying(80),
host character varying(31) DEFAULT '' NOT NULL,
insecure character varying(4),
"language" character varying(2),
mailbox character varying(50),
md5secret character varying(80),
nat character varying(5) DEFAULT 'no' NOT NULL,
permit character varying(95),
deny character varying(95),
mask character varying(95),
pickupgroup character varying(10),
port character varying(5) DEFAULT '' NOT NULL,
qualify character varying(3),
restrictcid character varying(1),
rtptimeout character varying(3),
rtpholdtimeout character varying(3),
secret character varying(80),
"type" character varying DEFAULT 'friend' NOT NULL,
username character varying(80) DEFAULT '' NOT NULL,
disallow character varying(100) DEFAULT 'all',
allow character varying(100) DEFAULT 'g729;ilbc;gsm;ulaw;alaw',
musiconhold character varying(100),
regseconds bigint DEFAULT 0::bigint NOT NULL,
ipaddr character varying(15) DEFAULT '' NOT NULL,
regexten character varying(80) DEFAULT '' NOT NULL,
cancallforward character varying(3) DEFAULT 'yes'
);

CREATE TABLE voicemail_users (
id serial NOT NULL,
customer_id bigint DEFAULT (0)::bigint NOT NULL,
context character varying(50) DEFAULT '' NOT NULL,
mailbox bigint DEFAULT (0)::bigint NOT NULL,
"password" character varying(4) DEFAULT '0' NOT NULL,
fullname character varying(50) DEFAULT '' NOT NULL,
email character varying(50) DEFAULT '' NOT NULL,
pager character varying(50) DEFAULT '' NOT NULL,
stamp timestamp(6) without time zone NOT NULL
);

CREATE TABLE queue_table (
name varchar(128),
musiconhold varchar(128),
announce varchar(128),
context varchar(128),
timeout int8,
monitor_join bool,
monitor_format varchar(128),
queue_youarenext varchar(128),
queue_thereare varchar(128),
queue_callswaiting varchar(128),
queue_holdtime varchar(128),
queue_minutes varchar(128),
queue_seconds varchar(128),
queue_lessthan varchar(128),
queue_thankyou varchar(128),
queue_reporthold varchar(128),
announce_frequency int8,
announce_round_seconds int8,
```

```
announce_holdtime varchar(128),
retry int8,
wrapuptime int8,
maxlen int8,
servicelevel int8,
strategy varchar(128),
joinempty varchar(128),
leavewhenempty varchar(128),
eventmemberstatus bool,
eventwhencalled bool,
reportholdtime bool,
memberdelay int8,
weight int8,
timeoutrestart bool,
PRIMARY KEY (name)
) WITHOUT OIDS;
ALTER TABLE queue_table OWNER TO asterisk;

CREATE TABLE queue_member_table
(
queue_name varchar(128),
interface varchar(128),
penalty int8,
PRIMARY KEY (queue_name, interface)
) WITHOUT OIDS;

--donem permisos a l'usuari asterisk
GRANT ALL ON TABLE cdr TO asterisk;
GRANT ALL ON TABLE extensions_conf TO asterisk;
GRANT ALL ON TABLE sip_conf TO asterisk;
GRANT ALL ON TABLE voicemail_users TO asterisk;
GRANT ALL ON TABLE queue_member_table TO asterisk;
GRANT ALL ON TABLE queue_table TO asterisk;
```

3)Configurar el postgresSQl per acceptar connexions de diferents màquines a la BD asterisk. Per fer-ho editem el fitxer de configuració ***/etc/postgres/8.1/main/pg_hba.conf*** tal com es mostra a continuació:

```
#TYPE  DATABASE        USER            [CIDR-ADDRES]           METHOD
local   asterisk        asterisk                                password
host    asterisk        asterisk        127.0.0.1/32            password
```

La primera línia permet connectar el localhost a la BD asterisk amb rol asterisk per mitjà d'un socket Unix amb el mètode d'autenticació password. La segona línia fa el mateix amb un socket TCP/IP.

4) Configurem asterisk perquè pugui accedir al PostgresSQL. Per fer-ho em de configurar dos fitxers, ***res_odbc.conf*** i ***extconfig.conf***.

```
;/etc/asterisk/res_odbc.conf
[ast_cnf]
dsn                     =>      asterisk
username                =>      asterisk
password                =>      asterisk
pre-connect             =>      yes
```

```
;/etc/asterisk/extconfig.conf
[settings]
sipusers =>     odbc,ast_cnf,sip_conf
sippeers =>     odbc,ast_cnf,sip_conf
iaxusers =>     odbc,ast_cnf,sip_conf
iaxpeers => odbc,ast_cnf,sip_conf
```

```
queues   => odbc,ast_cnf,queue_table
queue_members => odbc,ast_cnf,queue_member_table
```

5) Configurem els contexts que volem que vagin amb Realtime. Per fer-ho em d'editar el fitxer **/etc/asterisk/sip.conf** pels canals SIP o el fitxer **/etc/asterisk/iax.conf** pels canals IAX. Jo només ho faré pels canals SIP.

```
[ctx_realtime]
switch => Realtime
```

La primera línia indica l'inici d'un context anomenat ctx_realtime, la segona línia diem a asterisk que tots els canals d'aquest context són Realtime, i per tant ha d'anar a buscar-los a les taules que toquin, en el nostre cas a la taula **sip_conf.**

6) Insertar canals

```
insert into sip_conf(name,type,host,secret,context)
values('carles','friend','dynamic','psw_carles','carles_ctx');
```

7) Configurar el dial plan

```
insert into extensions_conf(context,exten,priority,app,appdata)
values('carles_ctx','1234',1,'Dial','SIP/carles');
```

CAPÍTOL 4-MISSATGERIA INSTANTÀNEA AMB JABBER

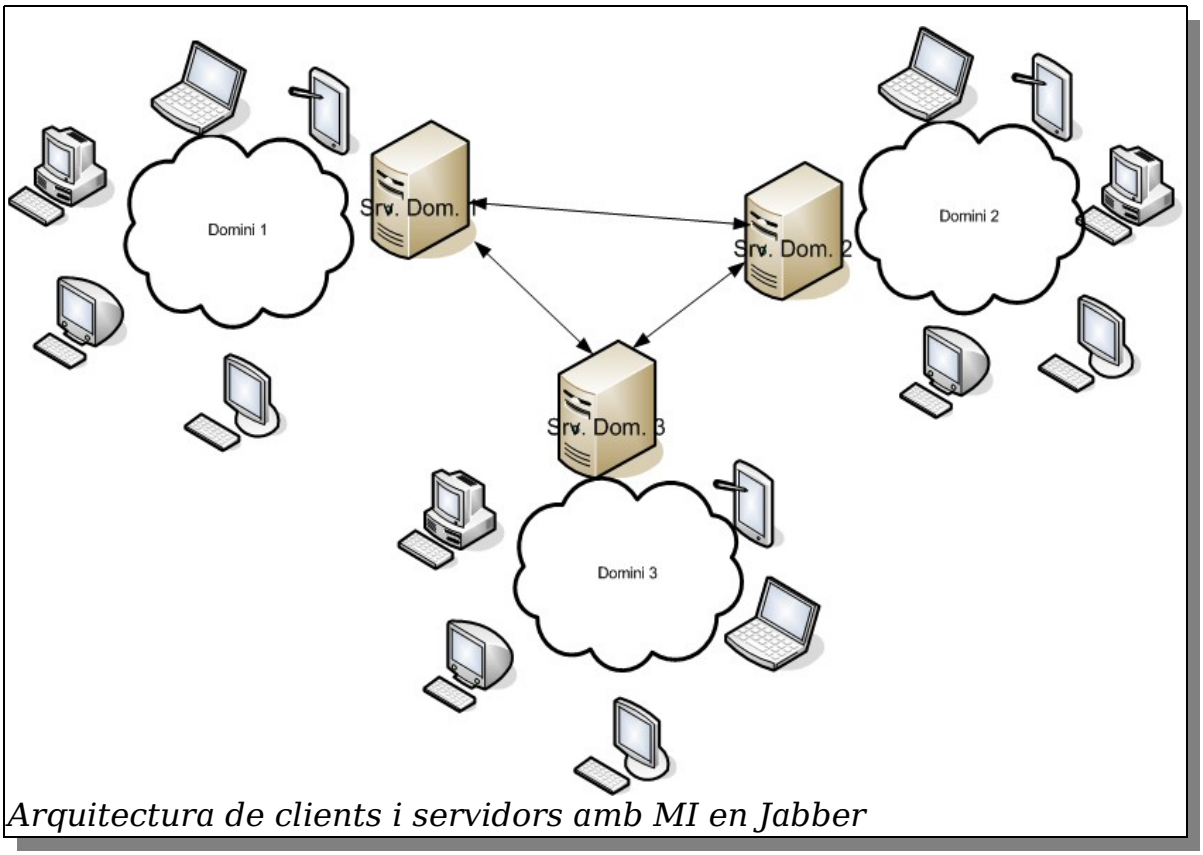
Jabber és un protocol de missatgeria instantània emprat en aquest projecte. Les característiques principals d'aquest protocol són:

- És obert.
- Segueix el model client-servidor.
- Està basat en missatges de text XML.
- En continua evolució.

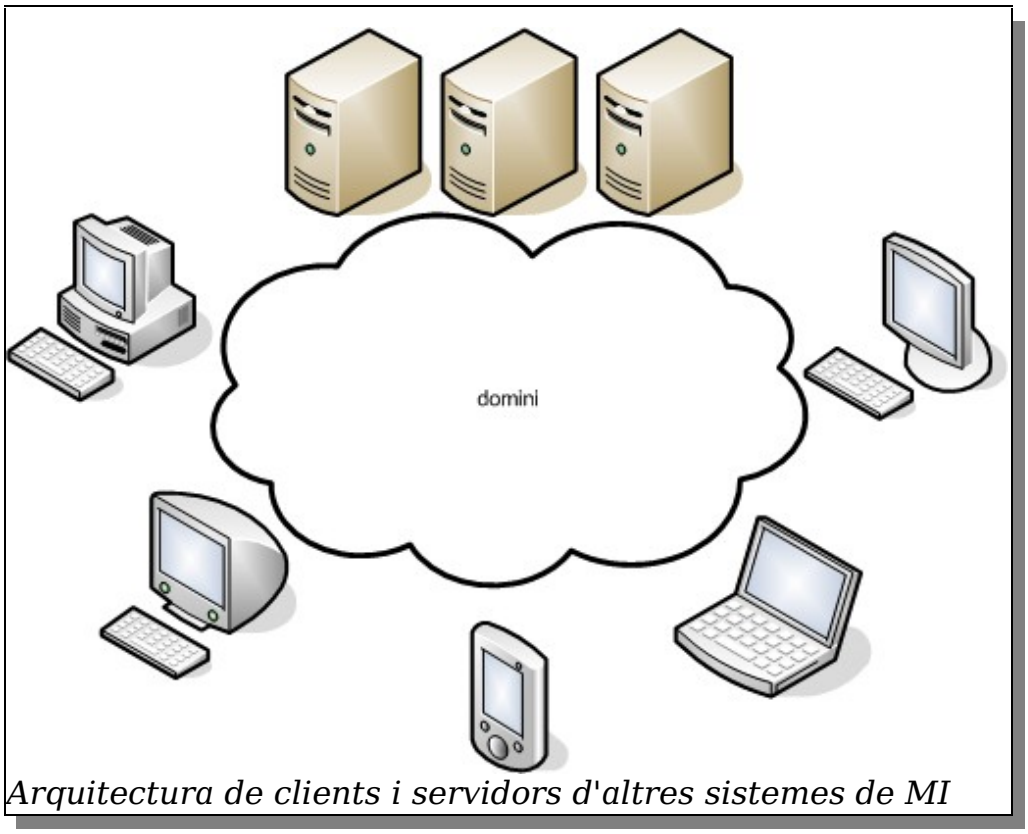
En aquest capítol farem una breu introducció a aquest protocol a fi d'obtenir els coneixements teòrics necessaris pel desenvolupament del projecte.

ARQUITECTURA

El protocol Jabber/XMPP segueix una arquitectura client-servidor. Els dominis Jabber/XMPP divideixen el mon en zones independents, cadascuna suportada i gestionada per un o més servidors del domini al qual pertany l'usuari. Aquesta característica difereix de la resta de sistemes de MI on un grup de servidors centralitzats gestiona tota la comunicació.



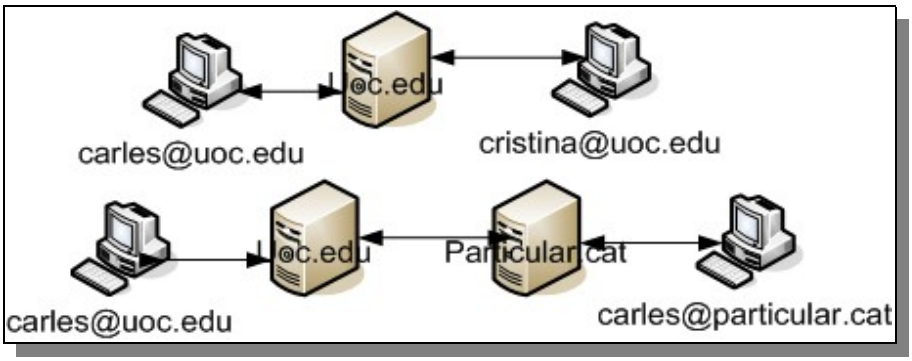
Arquitectura de clients i servidors amb MI en Jabber



Arquitectura de clients i servidors d'altres sistemes de MI

ESQUEMA DE FUNCIONAMENT

En Jabber els missatges passen del client origen al seu servidor de domini, i aquest el reenvia al servidor del domini de l'usuari destí el qual farà arribar el missatge al destinatari.



Jabber ID (JID)

Un Jabber ID o JID és el nom d'usuari emprat per accedir a un compte Jabber i la forma general és username@domini/recurs. La part del recurs és optativa i permet accedir al mateix compte des de diferents punts com

per exemple carles@uoc.edu/casa i carles@uoc.edu/feina. Si un usuari té dues sessions obertes, jabber envia el missatge al recurs més prioritari (indicat per l'usuari). Aquesta característica és un altre aspecte diferencial de Jabber respecte altres sistemes de MI. Els agents i altres elements automatitzats de la xarxa no necessiten tenir JID.

Exemple de funcionament

suposem que l'usuari usr1@uoc.edu vol enviar un missatge a l'usuari usr1@particular.cat, la seqüència d'esdeveniments seria la següent:

1. El client Jabber d'usr1@uoc.edu envia el missatge del servidor d'uoc.edu
   - Si el domini d'uoc.edu té bloquejat el domini particular.cat el missatge és rebutjat.
2. El servidor d'uoc.edu envia el missatge al servidor particular.cat
3. El servidor de particular.cat entrega el missatge a usr1@particular.cat
   - Si el domini particular.cat té bloquejat el domini uoc.edu el missatge és rebutjat.
   - Si usr1@particular.cat no està connectat el missatge és emmagatzemat per entregar-lo més tard.

Transports (gateway)

Una altra característica única en aquest protocol són els anomenats Transports, els quals permeten als usuaris accés a les xarxes emprant altres protocols, els quals poden ser altres sistemes de MI (MSN, AIM..) però també altres protocols com email o SMS. Aquest servei està present a nivell de servidors, i els clients Jabber només han de registrar en algun d'aquests gatways i proveir de la informació necessària per logarse en aquestes xarxes per disposar de comunicació en aquests sistemes.

Així doncs un client Jabber pot parlar amb altres sistemes de comunicació (MI i altres) sense tenir que conèixer tots aquests.

HTTP

Jabber permet als clients comunicar-se a través del servei http en aquells

casos que l'estructura de la xarxa ho requereixi, típicament usuaris darrera d'un firewall (el port http no el solen tancar). L'accés per mitjà d'HTTP es pot fer de dues maneres, per polling i per binding.

- Per polling els clients estan constantment enviant peticions (GET o POST) al servidor jabber preguntant-li si tenen un missatge nou.
- Per binding els clients fan us de connexions HTTP de llarga vida per rebre els missatges només quan algun altre client li envia. Aquest mètode és molt més eficient, que l'anterior.

Protocol de missatges

El protocol de missatges defineix l'estructura dels diferents missatges que gesitona Jabber. Consta de sis tipus de missatges:

- normal: missatges de correu.
- chat: missatges per la comunicació directa entre dos clients.
- groupchat: missatges dirigits a grups de persones
- headline: Usats per enviar informació genèrica de l'estat del sistema.
- error: missatges d'error.
- jabber:x:oob: missatges per enviament d'arxius entre clients.

L'estructura general d'un missatge és,

```
<message>
  <subject from='xxx' to='yyy' id='zzz' type=''></subject>
  <thread></thread>
  <body></body>
  <error></error>
</message>
```

Protocol de presencia

Aquest protocol és el responsable de gestionar la presencia de cadascun dels usuaris del sistema Jabber. La presencia no és limita a "connectat"/"no connectat" sinó que l'usuari pot definir els seus propis estats.

Una propietat d'aquest protocol és que un usuari que vulgui conèixer l'estat d'un altre usuari, el primer haurà de demanar permís al segon.

Aquest protocol és emprat en dos contexts:

- Actualització de la presencia: actualització de l’estat degut algun canvi d’estat.
- Subscripció de presencia: permet als usuaris subscriures a les actualitzacions de presencia d’altres usuari, així com controlar quins usuaris estan monitoritzant la meva presència.

En qualsevol cas és el servidor el que s'encarrega de fer arrivar als usuaris implicats els missatges corresponets.
Els diferents tipus de missatge de presència són:

tipus	***Descripció***
avaible	Missatge d'actualització que indica que l'usuari està preparat per rebre missatges.
unavailbe	Missatge d'actualització que indica que l'usuari no està disponible per rebre missatge.
subscribe	Missatge de petició de subscripció de presència, l'usuari que l'envia vol subscriures a la presència del destí.
unsubscribe	missatge de cancel·lació de subscripció de presència, l'usuari que l'envia vol cancel·lar la subscripció de presència del destí.
subscribed	Resposta que rebrà l'usuari que ha enviat un subscribe si aquesta ha estat aceptada pel destí.
unsubscribed	Resposta que rebrà l'usuari que ha enviat un unsubscribe o subscribe si aquesta ha estat denegada pel destí.
error	Indica problemes en la presència.
probe	missatge servidor-servidor que envia tota la informació de presència a un altre servidor.

Protocol de grup de xat

Aquest protocol s'encarrega de gestionar les converses en sales de xat, les quals s'identifiquen per un JID pròpi, i en les que podem realitzar quatre tipus d'operacions:

- Unirse al grup: envia un missatge del tipus “avaible” al grup.
- Enviar missatge a tots els membres: per fer-ho en el camp destí s'indica el JID de la sala. Previament s'ha d'haver unit.
- Enviar un missatge privat: per fer-ho en el camp destí cal posar

jid_sala/nickname.

- Abandonar grup: enviar un missatge “unavaible” al grup.

Protocol Info/Query

Aquest protocol s'encarrega de gestionar missatges per gestionar informació d'usuari. Els paquets IQ poden ser de dos tipus “get” o “set”, els primers per solicitar informació, mentre que els segons per posar-la. El format d'un missatge IQ és

```
<iq   type='set|get|result|error'
     to='jid_destino'
     from='jid_origen'
     id='unica'>
     <query xmlns='iq extension namespace'>
          <query_field1/>
          <query_field2/>
     </query>
</iq>
```

El tipus result és el resultat d'una petició set o get.
L'element <query> pot apareixer cap o N vegades, i generalment apareix en els missatges result, per retornar els valors d'una consulta. L'atribut xmlns permet definir namespace XML propis, els quals defineixen l'estructura de l'element query (els subelements) així com un protocol per tractar-los. Aquest és el mecanisme d'extensió de Jabber, i en l'actualitat ja existeix un gran nombre d'extensions, de les quals destaquem les següents:

- Registration protocol (jabber:iq:register): és l'usat per crear contes d'usuari i gestionar-les.
- Authentication protocol (jabber:iq:auth): Dedicat a la seguretat.

CAPÍTOL 5 -ESPECIFICACIÓ I DISSENY DE L'APLICACIÓ

En el capítols anteriors hem estudiat la base teòrica de les aplicacions de comunicació de missatgeria instantània així com de veu IP. També em vist com configurar una centraleta Asterisk que ens permeti gestionar una xarxa privada telefònica. En aquest capítol ens centrarem en l'anàlisi i disseny de l'aplicació que entengui ambdós sistemes de comunicació.

1 REQUERIMENTS

Es vol dissenyar una aplicació client que integri un sistema de comunicació instantània basat en el protocol Jabber i un sistema de veuIP, permeten així la comunicació oral (per mitjà d'un micro) i l' escrita.

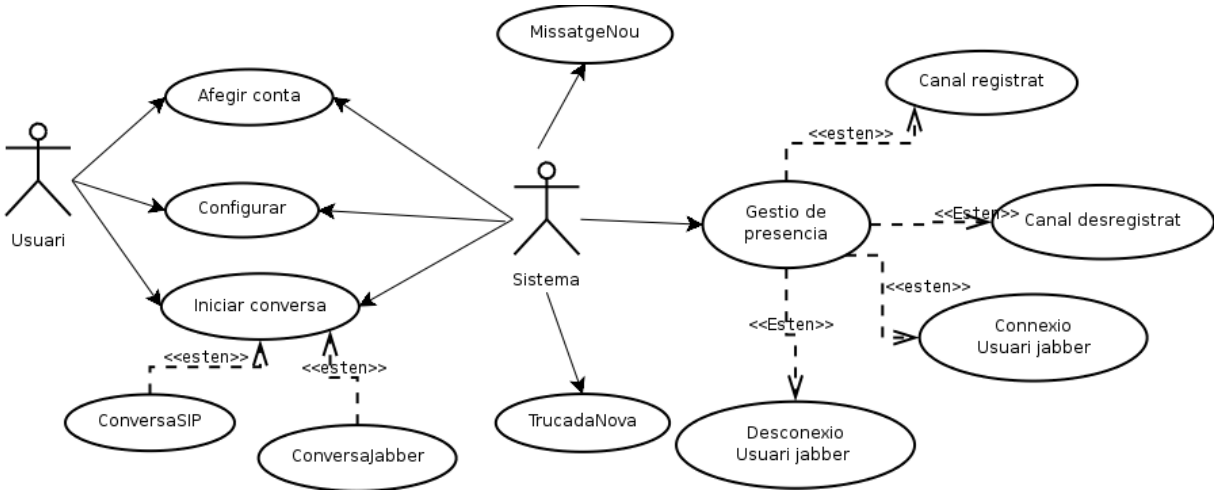
D'altra banda l'aplicació ha de poder crear contes d'usuari en i afegir-les al servidor jabber i a la centraleta asterisk aplicant-li una extensió.

Ha de permetre configurar els aspectes bàsics dels servidors: adreces ips, ports,..

Un altre requeriment d'aquesta aplicació és que analitzi els canvis d'estat dels usuaris dels dos protocols.

Per fer aquest apartat es seguira un ànalisi orientat a objectes amb notació UML, incloent-hi els diagrames més importants.

2 MODEL DE CASOS D'ÚS



Afegir Conta: crea una conta d'usuari nova.

Configurar: configura els paràmetres de l'aplicació.

Iniciar conversa: L'usuari vol comunicar-se amb algun altre usuari, per VeuIP o per XMPP.

TrucadaNova: és un cas d'ús de sistema que inicia aquest quan tenim una trucada nova.

Missatge nou: és un cas d'ús de sistema que inicia aquest quan rebem un missatge nou.

Gestió de presencia: Aquest cas d'ús s'inicia quan algún canvi d'estat relacionat amb la presència dels contactes succeeix.

3 MODEL CONCEPTUAL

En aquest apartat estudiarem les classes del model de domini de la nostra aplicació. La nostra aplicació integrarà tres dominis ben diferenciats:

- **Asterisk:** aquest gestionarà aquells aspectes relacionats amb la centraleta telefònica: crear comptes i assignar-los extensions, fer trucades,...
- **SIP:** aquest gestionarà aquells aspectes relacionats amb el protocol SIP: recepció de trucades, enregistrament al servidor registrar,...
- **Jabber:** Aquest gestionarà aquells aspectes relacionats amb el protocol XMMP: recepció de missatges, loggin al servidor, creació de comptes,...

El següent diagrama de classes UML podem veure les classes que incorporaran aquesta gestió així com les seves relacions.

Classes

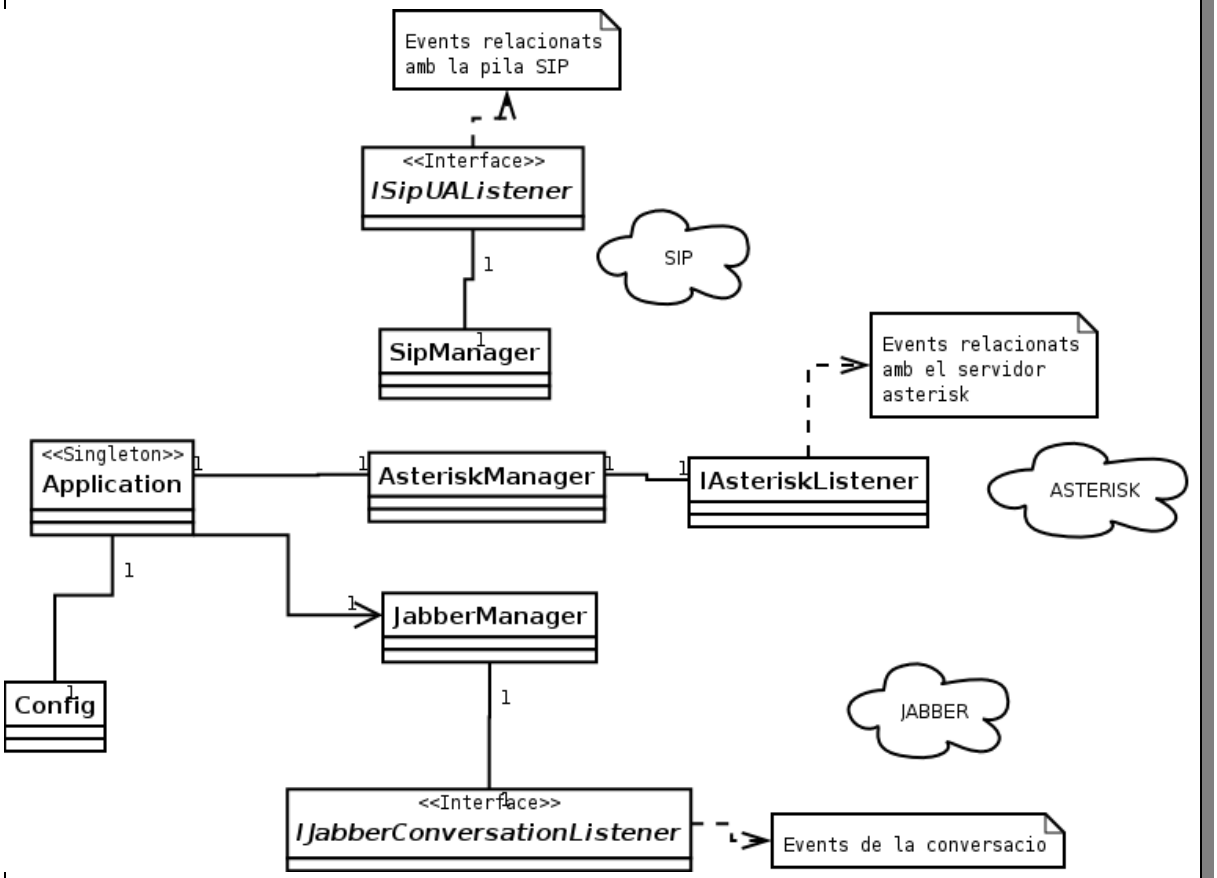


Diagrama conceptual de classes

Application

Aquesta classe és l'encarregada de publicar els mètodes visibles de l'aplicació, fent de façana. Implementa el patró singleton ja que només cal una instància. Té associacions amb els tres Managers dels quals en parlem a continuació, així com una associació amb una classe Config per accedir als paràmetres de configuració de la aplicació.

AsteriskManager

Classe que gestiona la interacció amb la centraleta asterisk i permet:

- crear comptes (canals) i assignar-los una extensió.
- obtenir la llista de connectats .

- rebre notificacions del servidor.

JabberManager

Aquesta classe gestiona la interacció amb el servidor Jabber i permet:

- crear comptes en el servidor Jabber.
- Enviar missatges a altres usuaris del mateix servidor.
- Obtenir la llista d'usuaris així com el seu estat.
- rebre notificacions del servidor

SipManager

Aquesta classe és una classe pont amb la pila SIP i permet:

- Realitzar trucades a altres clients sip.
- Rebre trucades.
- Enregistrar-se en algun servidor registrar.

Config

Aquesta classe conté els paràmetres de configuració de l'aplicació, i és l'encarregada de mantenir persistents aquests i recuperar-los quan es cau.

Interfícies

IsipUAListener, IjabberListener, IAsteriskListener

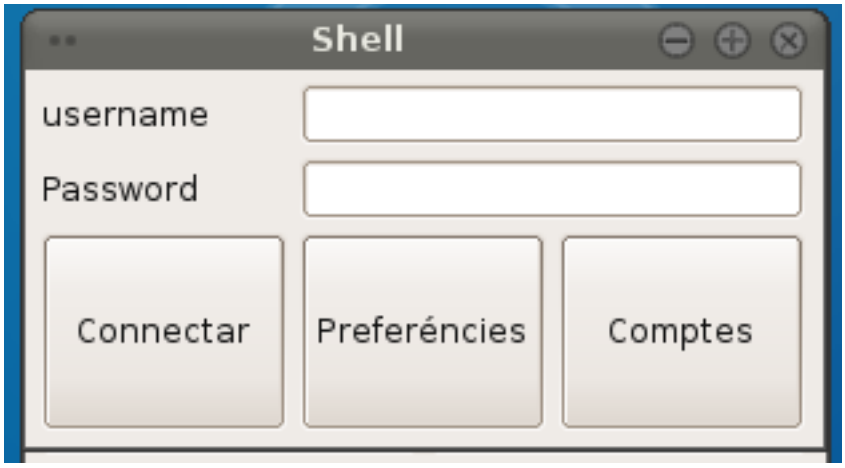
Les tres interficies incorporen els events de cadscun dels dominis (SIP,Asterisk i Jabber), i l'objectiu de les mateixes és desacoblar aspectes de la capa de presentació de la capa de la lògica del domini.

4 CAPA DE PRESENTACIÓ

Anem a veure quin aspecte tindran les pantalles de la nostra aplicació, així com les classes que l'implementen.

WinLogin

La primera vista és la d'entrada al nostre sistema, i presenta un aspecte força familiar en aquest tipus de programes. Des d'ella podem logarnos, accedir a la configuració de l'aplicació, o crear comptes nous.

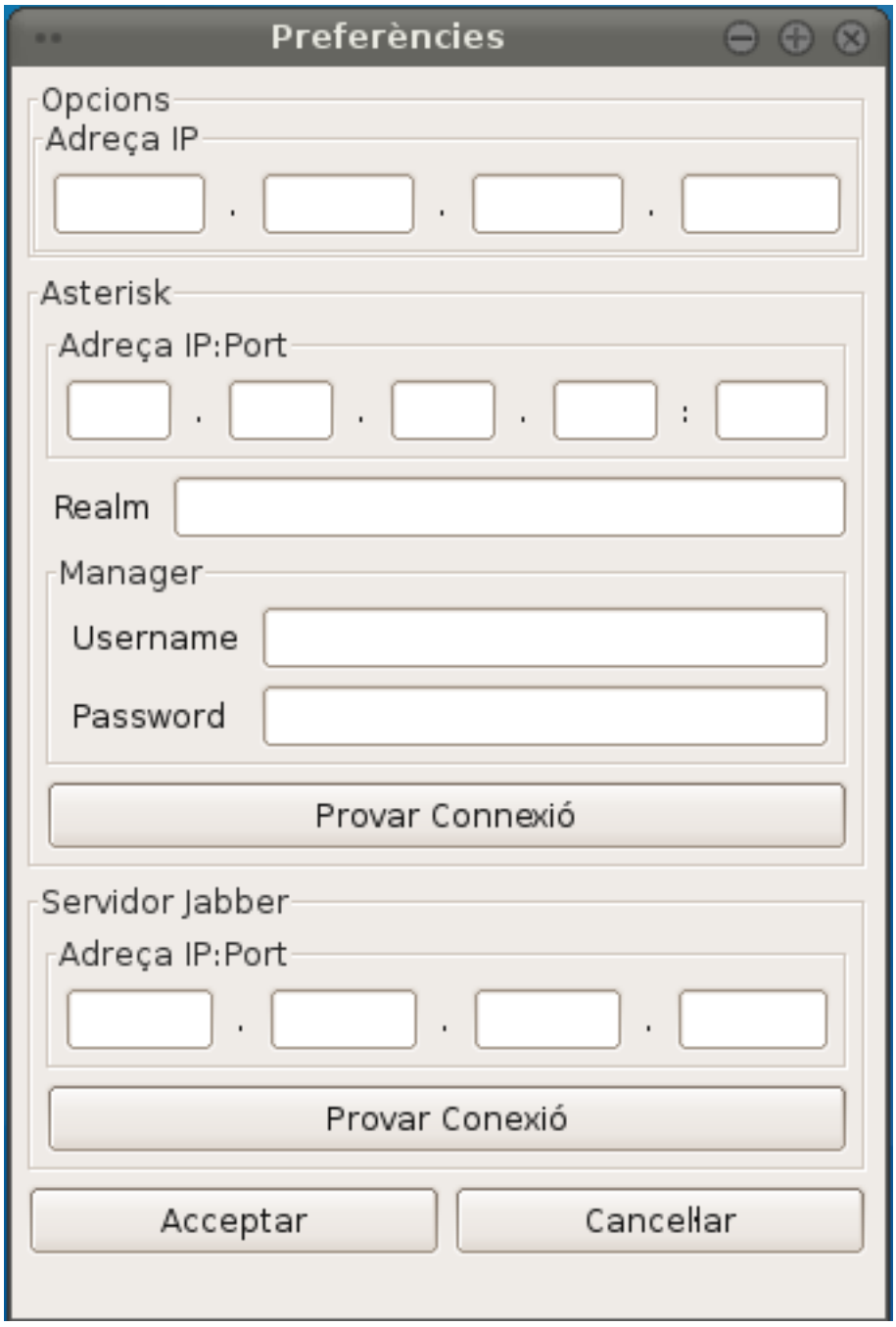


WinNewAccount

Aquesta vista és la que gestiona els nous comptes al nostre sistema.

WinPreferences

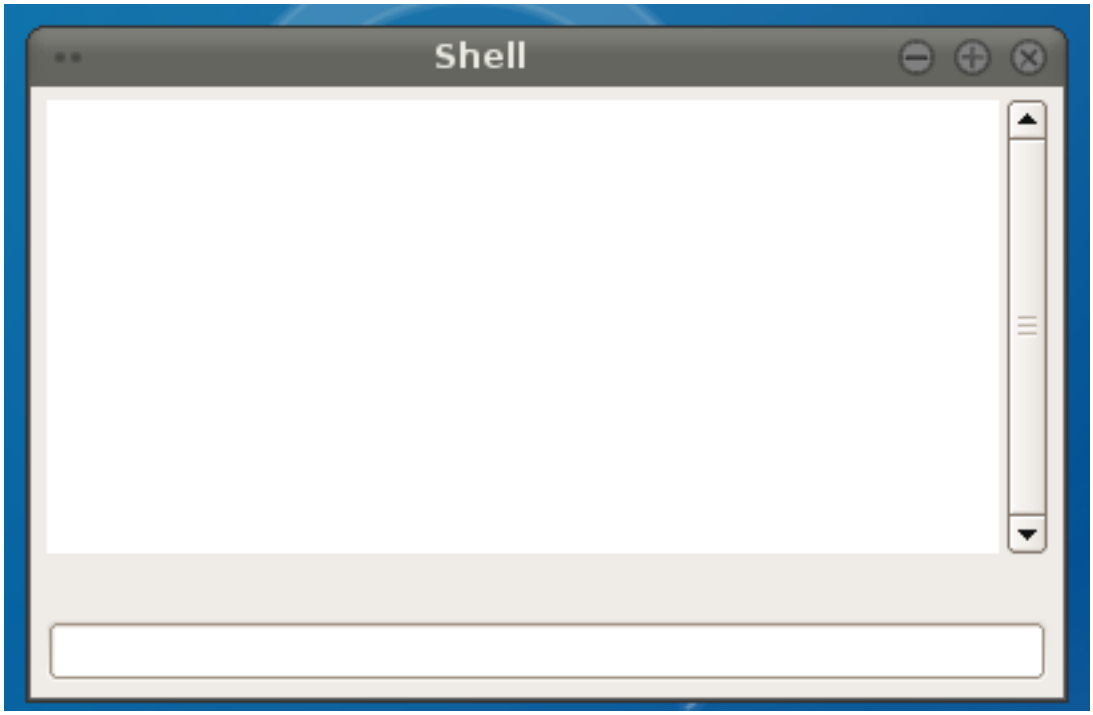
Aquesta vista permet configurar els paràmetres del sistema.


Preferències
Opcions
Adreça IP
Asterisk
Adreça IP:Port
Realm
Manager
Username
Password
Provar Connexió
Servidor Jabber
Adreça IP:Port
Provar Conexió
Acceptar
Cancel·lar

WinClient

La següent vista ens permet accedir als contactes de l'aplicació i iniciar converses per qualsevol dels dos sistemes.

WinJabberConversation



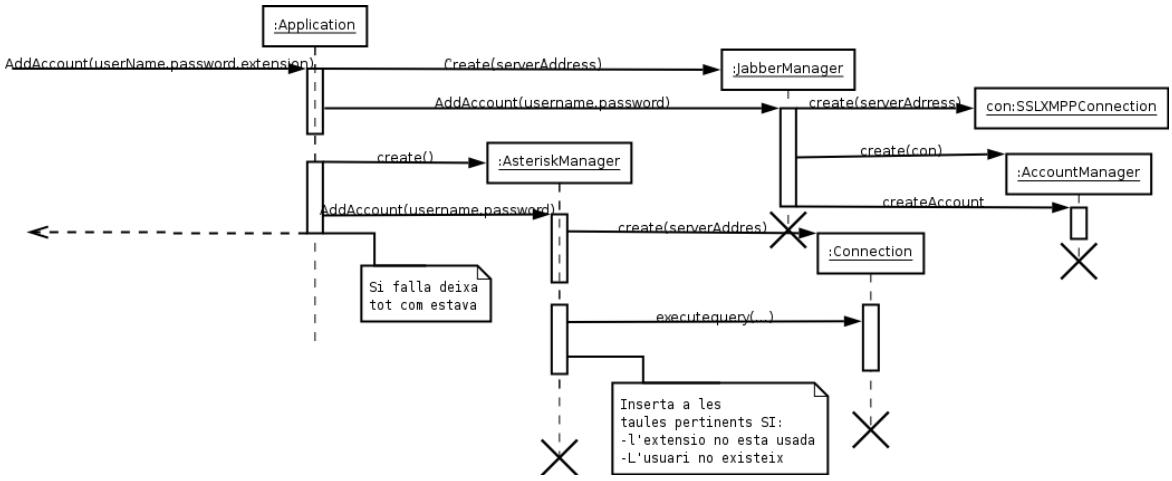
5 Diagrames de seqüència

En aquest apartat s'adjuntan alguns dels diagrames de seqüència més importants de la nostra aplicació.

AddAccount(username,password)

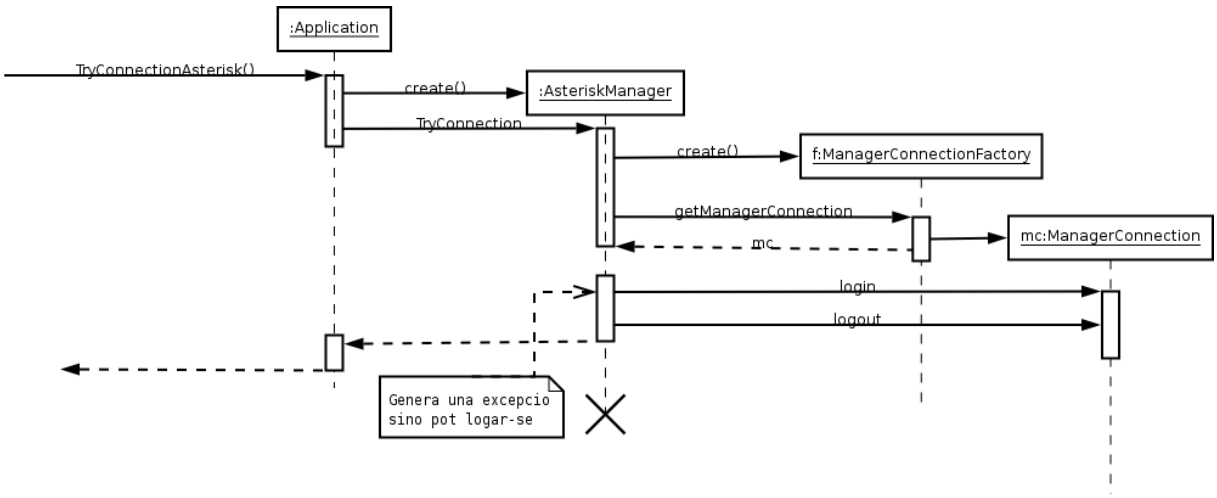
Aquesta operació crea un compte nou al servidor Asterisk i a Jabber si:

- El compte no està usat
- L'extensió no està usada



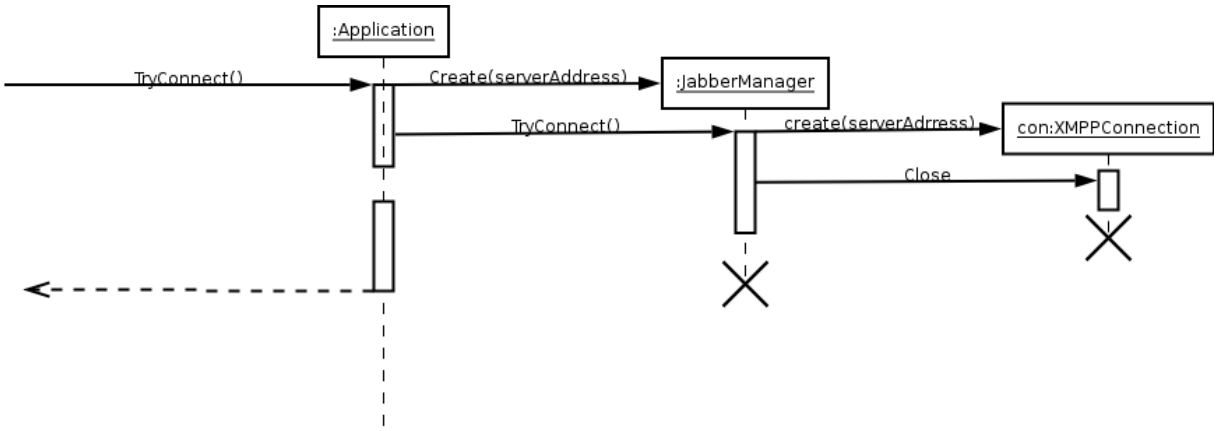
TryConnectionAsterisk

Aquesta operació prova connectar-se amb la centraleta asterisk



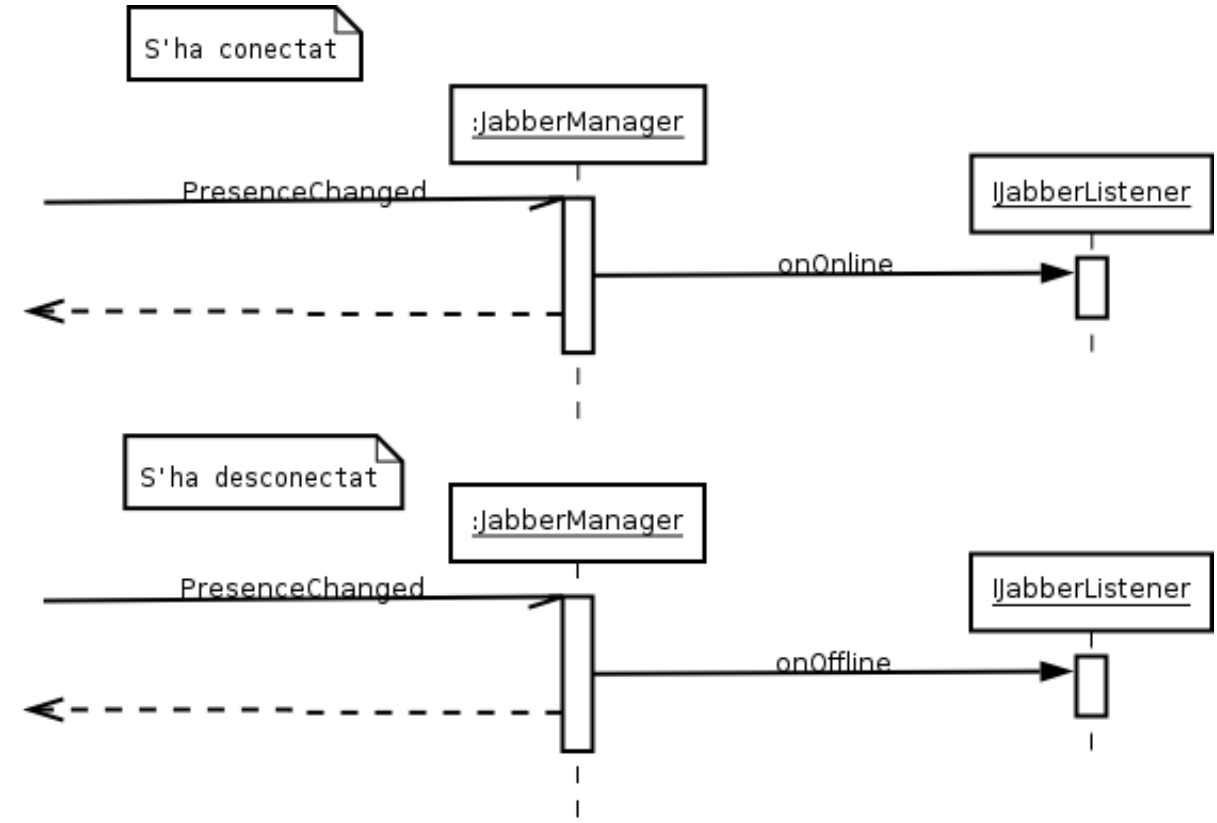
TryConnectionJanner

Aquesta operació intenta connectar-se amb el servidor Jabber



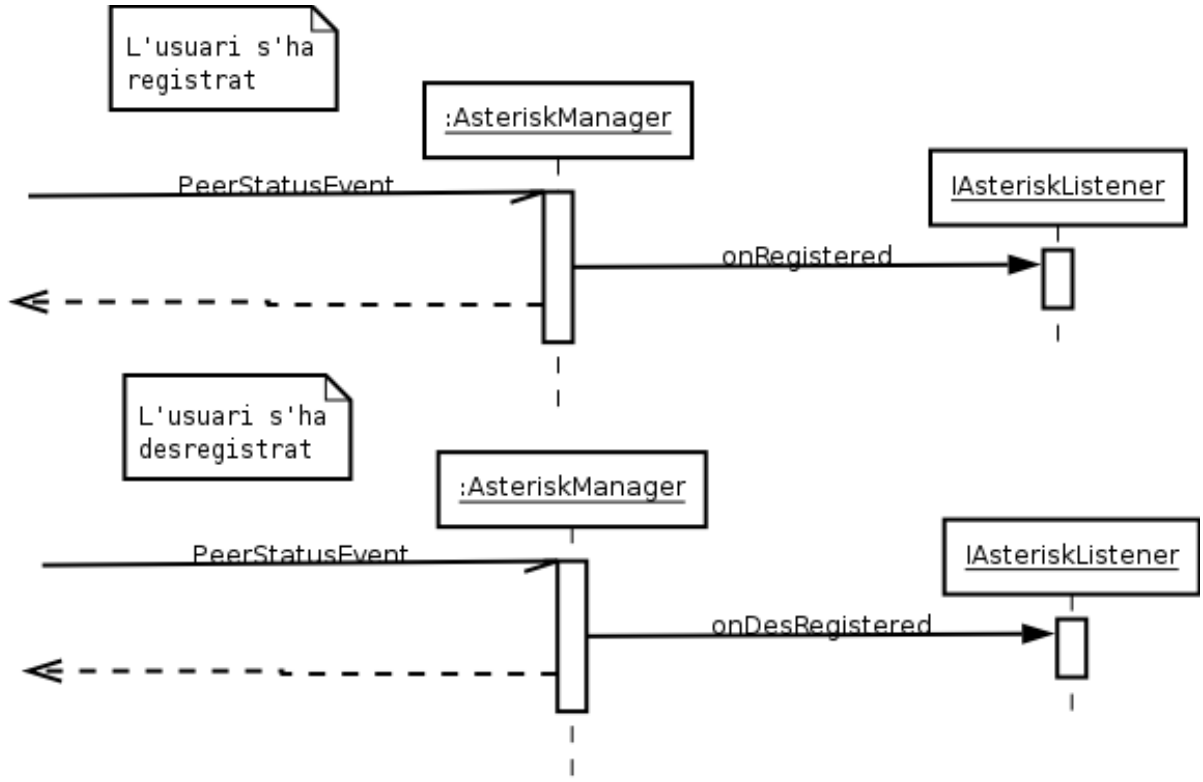
PrecenseChanged

Aquest és un event que envia el servidor Jabber quan la presència d'algun dels nostres contactes (de la llista de Roster).



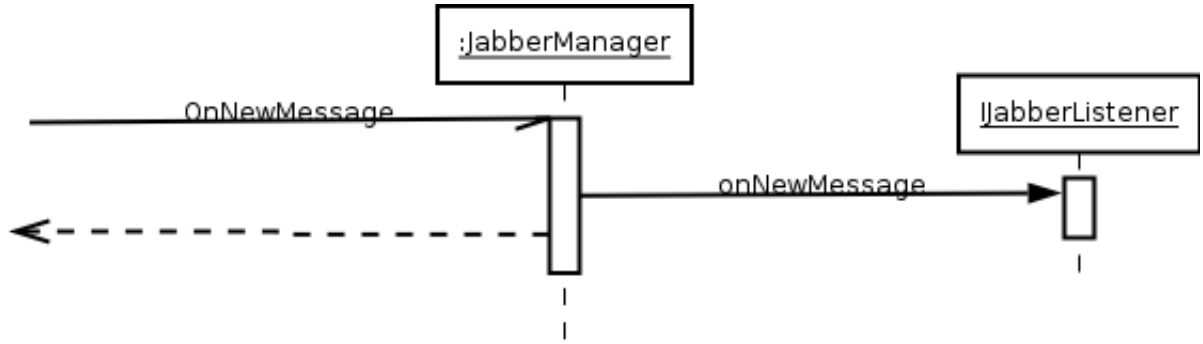
PeerStatusEvent

Aquest és un event de la centraleta asterisk que envia a l'aplicació quan algún canal s'ha registrat o desregistrat.



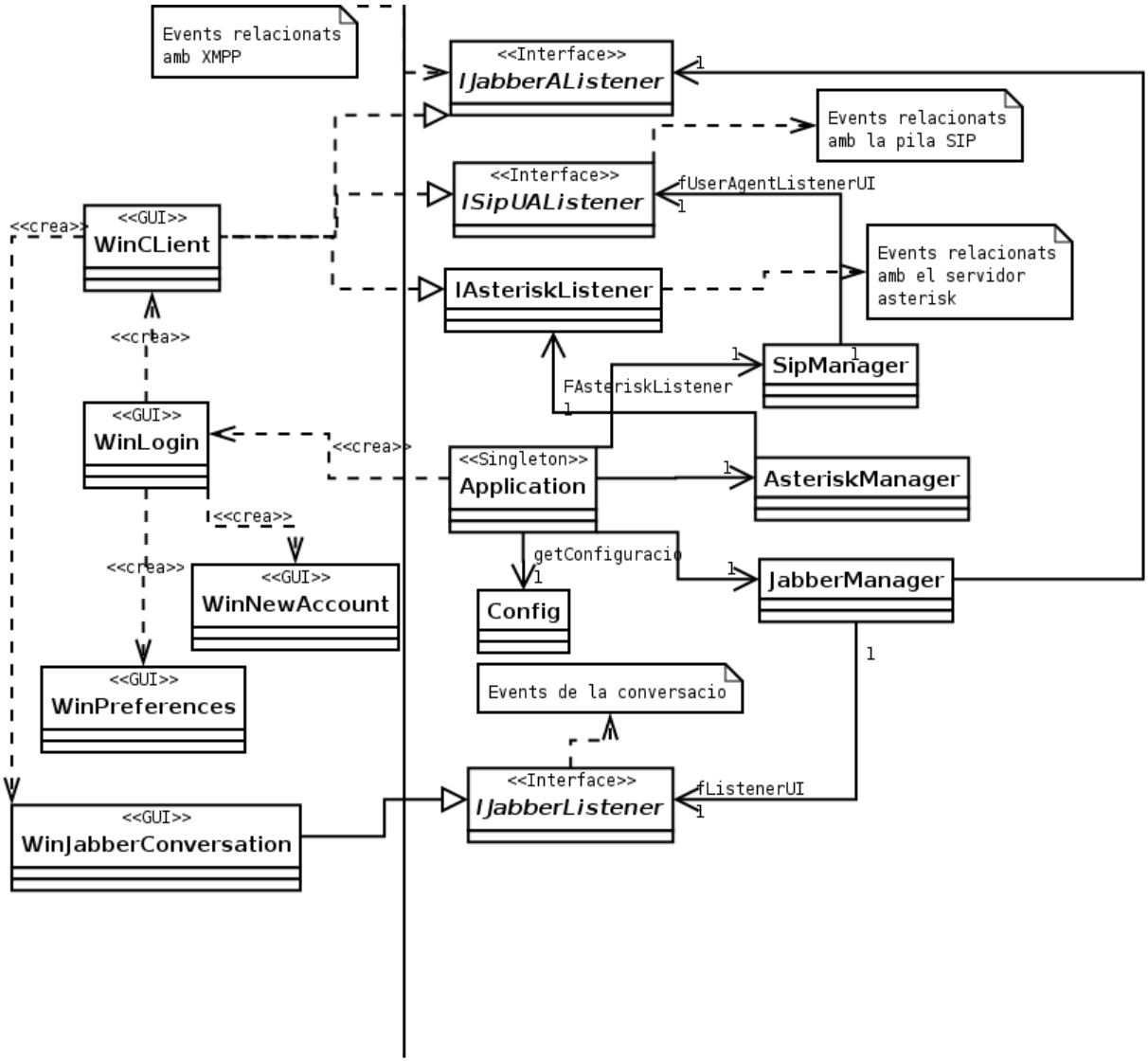
OnNewMessage

Aquest és un event que envia el servidor Jabber quan rebem un missatge nou.



6 Diagrama de les classes de disseny

Presentem a continuació el diagrama de les classes de disseny implementades així com les seves relacions.



CAPÍTOL 6 -CONCLUSIONS

1 Objectius coberts

L'objectiu principal d'aquest projecte s'ha cobert, es a dir el desenvolupament d'una aplicació que integrés sistemes de comunicació en temps real per internet, el qual permetés la comunicació en ambdós sistemes.

Per cobrir l'objectiu final s'ha hagut de fer un estudi dels sistemes subjacents per conèixer les seves característiques i els quals són: Asterisk com a servidor d'extensions, SIP com a protocol de veu IP, i XMPP com a protocol de missatgeria intantànea.

D'altra banda un cop s'han conegut les principals propietats d'aquests sistemes s'ha hagut d'analitzar diferents llibreries de programació per cadascún d'aquests.

Finalment s'ha fet l'anàlisi i desenvolupament de l'aplicació esmentada. Per fer-ho s'ha seguit una metodologia basada en Orientació a Objectes, emprant UML com a llenguatge de modelat.

2 Treballs futurs

Tot i l'assoliment dels objectius plantejats, cal dir que el producte obtingut és un prototip i no pas una aplicació d'usuari final. Això ha esdevingut així, en gran mesura a la dimensió dels requeriments i el temps estipulat per assolir-los (90 hores de treball) superat en escreix. És per això que aquest projecte ha obert una línia de desenvolupament de la qual és despren les següents ampliacions.

Gestió de grups

L'aplicació desenvolupada afegeix tots els contactes en un grup de contactes, encara que tant Jabber, per mitjà del Register protocol, com Asteris, per mitjà de “Contexts” permeten crear grups. Seria interesant incorporar aquesta carecterística en futures ampliacions.

Sistemes multiconversa

La comunicació tractada en l'aplicació és un a un, però existeixen altres modalitat on intervenen més de dos emisors/receptors. Aquestes modalitats en Asterisk es coneixen com sales de conversa, i en Jabber com chats i no s'han explotat en el sistema.

Enviament de fitxers

Aquesta carecterística permet enviar fitxers a través del protocol Jabber.

Gestió de la presència avançada

Jabber permet multiples estats de presència, a part de “connectat/desconnectat”, els quals no s'han pogut integrar. De fet, jabber permet a l'usuari definir els seus propis estats. Tot i això cal esmentar que aquesta carecterística no és fàcil, ja que el model de gestió de presència és força complex.

Integració de sistemes de video

Un altre carecterística interesant d'integrar en el nostre sistema és la capcitat de transmetre vídeo, el qual permetria obtenir un sistema de videotrucada real.

Integració a nivell de servidor

En aquesta aplicació s'han desenvolupat les clases de gestió dels sistemes subjacents i s'han integrat en una aplicació d'escriptori. Una futura ampliació seria moure aquestes classes (els Manager basicament) en un entorn servidor. Aquest efecte ens permetria dissenyar aplicacions de telefonia i MI molt lleugeres, podent-les implementar en entorns de recursos limitats com són els sistemes embeded (hardphone, pda,...).

Un altre entorn que podriem aprofitar aquesta característica seria el entorn web, oferint aquest sistema integrat en els navegador actuals. De fet, existeix un plugin per Mozilla anomenat MozPhone que es podria servir com a base de desenvolupament.

3 Planificació

A continuació presentem la planificació inicial, per comparar-la posteriorment amb la real.

Planificació *incial*

tasca	*data inici*	*data fi*	*Dies*
Estudi Asterisk			
-Instal.lació Asterisk i proves amb clients	4/3/2006	12/3/2006	9
-Estudi de funcionalitats i configuració d'aquestes	13/3/2006	19/3/2006	7
-Estudi de Llibreries programació	20/3/2006	26/3/2006	7
Protocols de Veu IP			
-Estudi Llibreries Veu Ip	15/4/2006	23/4/2006	9
Estudi Jabber			
-Instal·lació servidor jabber	24/4/2006	30/4/2006	7
-Estudi Protocol	1/5/2006	7/5/2006	7
-Estudi Llibreries de programació	8/5/2006	14/5/2006	7
Programació Aplicatiu			
-Interficie	15/5/2006	21/5/2006	7
-Integració	22/5/2006	4/6/2006	15
-Proves i correccions	5/6/2006	11/6/2006	7

Total: 82 dies

Planificació *final*

tasca	*data inici*	*data fi*	*Desviament*
Estudi Asterisk			
-Instal.lació Asterisk i proves amb clients	4/3/2006	12/3/2006	0
-Estudi de funcionalitats i configuració d'aquestes	13/3/2006	19/3/2006	0
-Estudi de Llibreries programació	20/3/2006	2/4/2006	+7
Protocols de Veu IP			
-Estudi Llibreries Veu Ip	15/4/2006	30/4/2006	+7
Estudi Jabber			
-Instal·lació servidor jabber	30/4/2006	6/5/2006	0
-Estudi Protocol	6/5/2006	13/5/2006	0
-Estudi Llibreries de programació	13/5/2006	21/5/2006	+1
Programació Aplicatiu			
-Interficie	22/5/2006	3/6/2006	+6
-Integració	3/6/2006	11/6/2006	-6
-Proves i correccions	12/6/2006	18/6/2006	0
			+15

Total: 82+15= 97 dies

El primer desviament, es produeix en l'estudi de les llibreries de programació Asterisk. Durant aquesta fase es descobreix que per disposar de configuració dinàmica d'aquest sistema (canviar els paràmetres del sistema) sense haber de reiniciar el servidor, cal emprar el que en terminologia Asterisk es coneix com REALTIME. Aquesta carecterística implica l'ús d'un SGBD (el postgreSQL), així com noves configuracions. Aquesta part no està massa ben documentada, i lo que hi ha està en versions anteriors del servidor (1.0), la qual cosa em va complicar força el desenvolupament. Al final amb l'ajuda dels forums es

va superar aquest entrebanc.
El segon desviament apareix novament en l'estudi de llibreries de programació de veuIP, degut en part a la complexitat d'aquest protocol, així com la comunicació amb asterisk. Cal dir que mirant aplicacions fetes amb les llibreries emprades s'aconsegueix solventar aquest entrebanc amb 7 dies de desviament.
El tercer desviament produit en l'estudi de les llibreries Jabber esdevé per la part de la gestió de la presència, que és l'aspecte més complex.
La programació de l'interficie retarda sis dies el nostre desenvolupament. Cal comentar que en aquesta etapa m'adono que l'eina emprada no és la més adient, però no disposo de marge suficient per corretgir-ho i segueixo endavant assumint el retard que provocarà. Aquesta eina és l'editor visual de l'eclipse, que en la versió 3.1 i la 3.2 (l'actual) és eïna molt pesat i que ralentitza molt el desenvolupament. En canvi s'avalua NetBeans (un altre IDE promocionat per Sun) i aqueta característica és molt àgil i més professional que el de l'eclipse. Tot i així no recondueixo l'aplicació perquè soc més partidari de les toolkits SWT que no pas Swing.
El darrer desviament es produeix en l'etapa d'integració però aquest cop és en negatiu, esdevingut en part per la pròpia pressió de l'apropament a la data final.
Així doncs és produeix un desviament total de dues setmanes.

BIBLIOGRAFIA

Protocols de VeuIP

http://es.wikipedia.org/wiki/Session_Initiation_Protocol

http://www.ietf.org/rfc/rfc3261.txt

Asterisk

Jim Van Meggelen, Jared Smith, and Leif Madsen. *The asterisk HandBook Version 2 Draft.* O'really, 2005.

Mark Spencer, Mack Allison, Christopher Rhodes, The Asterisk documentation Team. *Asterisk, the future of telephony.* Digium 2003.

http://forums.digium.com/

http://www.voip-info.org/wiki-Asterisk

http://www.asteriskdocs.org/modules/news/

http://www.asteriskguru.com/

Jabber

http://www.xmpp.org/specs/

http://www.jabberes.org/ejabberd-basico

http://www.jivesoftware.org/smack/

JAVA i altres tecnologies

http://java.sun.com/docs/books/tutorial/

http://jdbc.postgresql.org/

http://www.eclipse.org

http://eclipse-plugins.2y.net/eclipse/plugins.jsp