UNIVERSITÀ DEGLI STUDI DI PARMA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Dottorato di Ricerca in Tecnologie dell'Informazione
XX Ciclo

Lorenzo Lazzari

SISTEMI MULTI-AGENTE E STRUMENTI MULTIMEDIALI AL SUPPORTO DI ATTIVITÀ COLLABORATIVE

DISSERTAZIONE PRESENTATA PER IL CONSEGUIMENTO
DEL TITOLO DI DOTTORE DI RICERCA

Gennaio 2008

Ringraziamenti

Ringrazio di cuore la mia famiglia per essermi stata sempre accanto ed avermi supportato anche durante quest'esperienza post-laurea.
Ringrazio il mio tutor di dottorato Prof. Ing. Agostino Poggi per l'assistenza, la disponibilità e le doti scientifiche ed umane dimostrate in questi anni.
Ringrazio con affetto i colleghi del laboratorio AOT: Alessandro, Marco, Federico, Michele e Paola.
Ringrazio infine tutti coloro che mi sono stati vicini nel corso di questi ultimi anni.

Indice

Prefazione

Il grande progresso dell'informatica e delle telecomunicazioni che ha caratterizzato questi ultimi anni ha ormai assunto i connotati di una vera e propria rivoluzione sociologica, coinvolgendo un numero sempre maggiore di realtà. In questa grande trasformazione, due termini, in particolare, hanno rivestito un ruolo molto importante, incrementando in modo esponenziale la propria diffusione: multimedialità ed Internet. La possibilità di utilizzare l'infrastruttura offerta da Internet per consentire la comunicazione mediante flussi di diversi tipi (audio/video, ma anche flussi di dati in genere), nel corso degli anni ha fatto del Web un potente strumento adatto a garantire una comunicazione o, più in generale, un'interazione multimediale fra soggetti fisicamente lontani, permettendo loro di prendere parte in modo attivo a sessioni di lavoro collaborativo.

Parallelamente alla diffusione di strumenti basati sul Web al supporto di attività collaborative di gruppi di lavoro virtuali, gli ultimi anni hanno visto lo sviluppo di sistemi distribuiti intelligenti basati sulla tecnologia ad agenti. Con il termine agente intendiamo un'entità computazionale in grado di percepire e comportarsi all'interno di un ambiente. Ciò significa che un agente non è un'entità isolata, ma può comunicare e collaborare con altre entità. Impiegati singolarmente, gli agenti sono destinati a diventare virtualmente inutili, tant'è che usualmente si parla di sistemi multi-agente, sistemi software composti da un numero anche elevato di agenti che svolgono in modo coordinato un insieme di attività finalizzate al raggiungimento di un obiettivo comune. Agenti sono oggi utilizzati, ad esempio, per la ricerca su Internet di informazioni con contenuto semanticamente rilevante per l'utente.

Sistemi al supporto di attività collaborative e tecnologia multi-agente rappresentano il punto di partenza dell'attività di ricerca descritta nel presente lavoro di tesi. L'attività si articola su diversi fronti, presentando anche aspetti del tutto innovativi che coinvolgono una tecnologia in grande ascesa, la televisione digitale (o DTV) basata su standard MHP. Da alcuni anni a questa parte, stiamo infatti assistendo ad una veloce migrazione dalla classica TV analogica verso questo nuovo paradigma di trasmissione, che porta principalmente due grandi vantaggi: l'opportunità di trasmettere in broadcast diversi canali nella medesima banda e, in particolare, la possibilità di inviare applicazioni software interattive parallelamente ai contenuti audio/video. Questi due aspetti hanno portato alla diffusione su larga scala di questo tipo di tecnologia, che si sta imponendo come un importante mezzo per lo sviluppo di nuove tipologie di servizi. Da queste considerazioni prende il via un'attività di ricerca finalizzata all'integrazione dei già potenti mezzi messi a disposizione dalla TV digitale con la tecnologia multi-agente, nell'ottica di fornire alla fascia di utenza della DTV una serie di servizi evoluti di collaborazione e di interazione diretta.

Possiamo dunque suddividere il percorso di ricerca in tre macro-aree e di conseguenza la presente tesi si struttura in tre capitoli, ognuno incentrato su una particolare tecnologia: nel capitolo 1 viene trattato l'argomento dei sistemi multi-agente, il capitolo 2 è incentrato sui sistemi di comunicazione multimediale su IP ed infine il capitolo 3 tratta l'argomento della tecnologia DTV-MHP. Ogni capitolo presenta una descrizione teorica dell'argomento e gli aspetti pratici ed implementativi oggetto della relativa attività di ricerca.

Come si noterà, la distinzione concettuale fra i tre capitoli non è così netta. In particolare, la tecnologia multi-agente presentata nel primo capitolo viene ripresa anche nelle fasi pratiche di sviluppo e di implementazione software descritte nei capitoli successivi. Questa peculiarità caratterizza l'intera attività di ricerca, un percorso trasversale che coinvolge filoni di ricerca eterogenei ma accomunati dallo sviluppo di servizi al supporto di attività collaborative di comunità di utenti.

1
Sistemi Multi-Agente

Questo capitolo presenta l'attività di ricerca incentrata sui sistemi multi-agente. Sulla tecnologia ad agenti, descritta inizialmente da un punto di vista teorico, si basa la fase implementativa finalizzata allo sviluppo di RAVE, una piattaforma per la creazione di sistemi multi-agente complessi ed accessibili da interfaccia Web, e di due sistemi, RAP e CAFE, al supporto di comunità di utenti.

1.1 Introduzione ai Sistemi Multi-Agente

Nel campo della computer science, un agente rappresenta un sistema software capace di azioni autonome orientate al raggiungimento di un obiettivo prefissato. Quella degli agenti è una tecnologia traversale utilizzabile in diverse aree: dall'ingegneria del software all'intelligenza artificiale.

Possiamo definire un agente come una particolare tipologia di astrazione software, nello stesso modo in cui lo sono i metodi, le funzioni, e gli oggetti. Un oggetto è un'astrazione ad alto livello che descrive metodi e attributi di un componente software, un agente offre un modo conveniente e potente per descrivere un'entità software complessa. Piuttosto che essere definita in termini di metodi e attributi, un agente è definito in termini di comportamenti (behaviour).

Dal punto di vista funzionale, un agente è un sistema a cui un utente (o un'altra componente software) delega lo svolgimento di uno o più task per il raggiungimento di un particolare goal. Questo meccanismo di delega implica che

un agente sia in grado di prendere decisioni in modo autonomo, mostrando, per certi versi, caratteristiche di intelligenza artificiale che gli consentono di assistere l'utente del sistema e, in particolare, di adattarsi al suo comportamento pur continuando nel contempo ad eseguire task ripetitivi.

In questo senso nella letteratura scientifica molto spesso gli agenti sono chiamati "autonomous intelligent agents", sottolineando il fatto che si tratta di entità indipendenti ed in grado di adattarsi all'evolversi delle circostanze. Al termine "agent" vengono molto spesso associati termini quali "autonomous", "proactive", "intelligent" o "social software components".

Un sistema che vede l'interazione fra diversi agenti viene detto "sistema multi-agente" (o MAS, Multi-Agent System). Tipicamente i singoli agenti che compongono un MAS non possiedono tutti i dati o i metodi necessari per raggiungere un determinato obiettivo, ma hanno la necessità di collaborare con gli altri agenti del sistema. Un MAS è dunque un sistema composto da diversi agenti che, complessivamente, hanno la capacità di raggiungere obiettivi che sarebbero molto difficilmente raggiungibili da un singolo agente o da un sistema monolitico.

Sebbene i sistemi multi-agente siano spesso caratterizzati come un argomento prettamente di ricerca, alcuni progetti di ricerca [8] hanno dimostrato la loro applicabilità in contesti reali comprendenti ad esempio attività collaborative, trasporti, logistica e altri campi. I sistemi multi-agente, per le proprie caratteristiche intrinseche, trovano inoltre quale campo naturale di applicazione il networking. Affidare la gestione di una rete a sistemi autonomi e proattivi come i MAS consentono infatti di ottenere ottimi risultati in termini di load balancing, di scalabilità e di robustezza della rete stessa.

Nel corso degli ultimi vent'anni hanno preso il via diversi progetti di ricerca, spostandosi sempre di più dal piano teorico ad applicazioni e tool pratici. Ciò che emerge è che la tecnologia multi-agente, quando propriamente ingegnerizzata ed integrata con altre tecnologie emergenti, consente di ottenere un valido e potente approccio nella risoluzione di diversi problemi.

Oltre ai campi di applicazione citati in precedenza, l'utilizzo della tecnologia multi-agente consente di ottenere vantaggi significativi in altre importanti aree di

ricerca e di sviluppo quali l'autonomic computing, il grid computing, il pervasive computing, i sistemi di simulazione ed, in particolare, i Web services ed il service-oriented computing. Anche se molti di questi campi sono direttamente o indirettamente collegati ad Internet e al Web, il mondo della ricerca non ha ancora investigato sulla possibilità di un'effettiva integrazione fra sistemi multi-agente ed applicazioni Web. Come sarà possibile apprendere in questo capitolo, l'integrazione di queste due tecnologie presenta sicuramente aspetti molto interessanti che saranno valutati sia da un punto di vista teorico sia sul lato implementativo tramite la presentazione di alcune applicazioni pratiche.

1.2 Tecnologie Web e Sistemi Multi-Agente

La ricerca sui sistemi multi-agente ha fornito negli ultimi anni risultati molto interessanti ed ha posto solide basi teoriche grazie alle quali i sistemi multi-agente oggi rappresentano una tecnologia decisamente consolidata e credibile. Il passo successivo è rappresentato quindi dall'applicazione di tali basi teoriche, ossia dall'utilizzo degli agenti all'interno di sistemi reali. Come anticipato nella sezione precedente, la tecnologia multi-agente ha già trovato una concreta applicazione, sia all'interno di progetti di ricerca sia da parte di aziende IT, in sistemi di supporto ad attività collaborative, software per la gestione di logistica, applicazioni grafiche ed altri campi. Dato il crescente interesse verso lo sviluppo di applicazioni e business Web-based, sembra abbastanza naturale che il passo successivo consista nello sviluppo di nuove tecnologie e framework che consentano di sfruttare le potenzialità dei sistemi multi-agente all'interno di applicazioni Web. Nonostante questo trend, non sono ancora rintracciabili veri e propri filoni di ricerca che si occupino nello specifico di questo argomento.

L'attività di ricerca descritta in questo capitolo è appunto incentrata su questo problema e si basa, in termini applicativi, sullo sviluppo di un framework del tutto innovativo per l'implementazione di applicazioni basate contemporaneamente su tecnologie Web e sistemi multi-agente.

1.2.1 JADE

JADE (Java Agent DEvelopment Framework) [29] è probabilmente il più famoso ed utilizzato ambiente di sviluppo per sistemi ad agenti. L'obiettivo di questo framework software è consentire al programmatore un'agevole implementazione di sistemi ad agenti tramite l'utilizzo di un middleware conforme alle specifiche FIPA [18] ed un set di tool grafici di grande aiuto nelle fasi di sviluppo e di debugging. JADE, realizzato interamente in Java da parte dell'Università degli Studi di Parma per conto di Telecom Italia Lab, rappresenta quindi una piattaforma altamente ingegnerizzata per la creazione e la gestione di sistemi ad agenti.

La caratteristica principale di JADE è sicuramente rappresentata dalla struttura distribuita del sistema stesso, che rende possibile la dislocazione della piattaforma su una rete di calcolatori eterogenei (workstation e personal computer con diverso sistema operativo, ma anche computer palmari) e comunicanti attraverso rete fissa o mobile. Inoltre, come detto pocanzi, JADE rende disponibili componenti software per realizzare agenti che comunicano seguendo lo standard dettato dal linguaggio FIPA ACL e semplifica notevolmente la vita del programmatore fornendogli agevoli strumenti grafici di sviluppo. Inoltre l'adozione del linguaggio Java come linguaggio di programmazione semplifica notevolmente l'integrazione dei sistemi ad agenti con le tecnologie di rete. JADE è distribuito in licenza open-source LGPL da Telecom Italia Lab ed è già stato ed è attualmente utilizzato in diversi progetti di ricerca nazionali ed internazionali.

L'implementazione di tutti i sistemi multi-agente descritti nel presente lavoro di tesi si basa sull'utilizzo di JADE.

1.2.2 JADE e Tecnologie Web

I sistemi che saranno descritti nelle seguenti sezioni presentano due componenti distinte: il "cuore" affidato ad un sistema ad agenti sviluppato tramite JADE ed una parte Web che rappresenta l'interfaccia fra il sistema stesso e l'utente o la comunità di utenti che lo utilizzano. Nella descrizione verrà dato maggiore risalto al primo

aspetto, dal momento che l'attività di ricerca ha riguardato in modo particolare la progettazione dei sistemi ad agenti. Per completezza, in questa sezione si parla anche brevemente della parte Web, la cui implementazione si basa sulla tecnologia JSP (Java Server Pages), che rende possibile l'inclusione di contenuti dinamici e/o codice Java all'interno di documenti HTML.

Come sarà possibile notare nei sistemi descritti in seguito, l'integrazione fra agenti e pagine JSP si basa sullo sviluppo di un particolare tipo di agente che funge da proxy fra le due parti, una sorta di livello intermedio per la sincronizzazione e lo scambio di informazioni fra la parte Web JSP-based ed il sistema multi-agente.

1.3 RAVE - Remote Assistance in a Virtual Environment

RAVE [33], acronimo di "Remote Assistance in a Virtual Environment", rappresenta una piattaforma su cui costruire sistemi multi-agente complessi ed accessibili da interfaccia Web al supporto di gruppi di utenti impegnati in progetti comuni o interessati ad un medesimo argomento. L'intento è quello di fornire agli utenti del sistema una serie di strumenti di assistenza e di reperimento di informazioni, una sorta di "assistenza in linea" per risolvere nel più breve tempo possibile problemi legati al reperimento di informazioni specifiche. La piattaforma è composta da diversi moduli base o di utilità generale, che possono essere utilizzati ed affiancati da altri moduli specifici nel corso della particolare implementazione, come sarà poi descritto nella sezione 1.4, in cui si presenta il primo sistema basato su RAVE, chiamato RAP (Remote Assistant for Programmers) e finalizzato al supporto di comunità di programmatori Java.

Un'applicazione RAVE è generalmente organizzata come una rete dinamica composta da diversi peer, ognuno dei quali rappresentato da una piattaforma multi-agente distribuita a sua volta su uno o più nodi computazionali e contenente gli agenti preposti a gestire l'accesso alle informazioni presenti su quel nodo. Un'importante caratteristica della piattaforma consiste nel fatto che RAVE

consente l'agevole sviluppo di applicazioni distribuite per la gestione di servizi del tutto eterogenei. Questi servizi possono essere volti sia alla gestione dell'interazione diretta tra gli utenti del sistema sia al supporto all'accesso degli utenti ai documenti ed alle informazioni già presenti e catalogate. Inoltre ogni servizio può essere affidato a diversi "service manager", vale a dire che uno stesso servizio può ad esempio gestire e ritornare all'utente informazioni provenienti da diversi repository. I concetti appena presentati sono schematizzati in figura 1.1, in cui una richiesta di un utente (query) indirizzata ad una piattaforma viene prima inoltrata al servizio corretto e successivamente al service manager corretto.

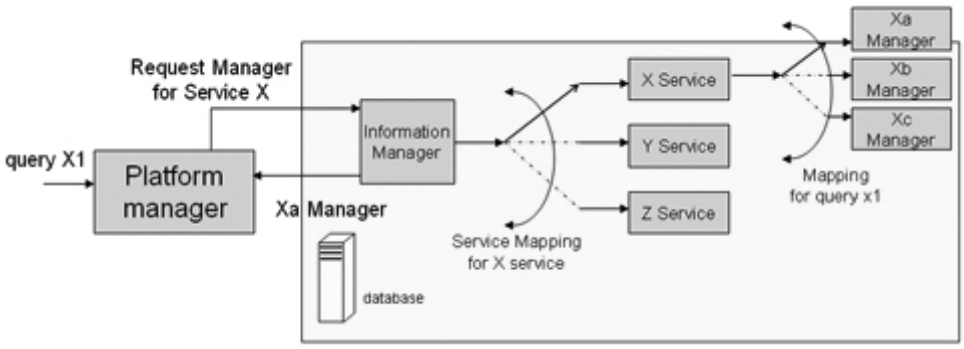

Figura 1.1 - Esempio di gestione di una query in RAVE

Prima di scendere nello specifico presentando in modo più dettagliato gli aspetti principali di RAVE, vale la pena sottolineare che RAVE è stato progettato per sfruttare la tecnologia ad agenti come strumento per lo sviluppo di applicazioni: la natura del progetto prettamente volta al reperimento intelligente di informazioni rappresenta solo l'idea iniziale, che si concretizza nella prima implementazione di RAVE, RAP, ma, come si vedrà in seguito con il sistema CAFE, il framework consente lo sviluppo di applicazioni e di servizi del tutto eterogenei.

1.3.1 Il Web Agent

L'utilizzo di un canale diretto di comunicazione tra un'applicazione Web ed uno o più agenti del sistema rappresenta senz'altro la soluzione più agevole per

garantire l'interazione utente-agente, ma comporta il grande svantaggio di collegare il sistema multi-agente con un'applicazione esterna. Pertanto, il progetto RAVE è partito con l'aggiunta di un layer di comunicazione tra la parte Web e gli agenti. Lo strato, rappresentato anch'esso da un agente, è stato chiamato "Web Agent". La soluzione presenta un approccio simile a quello accennato nella sezione 1.2.2 relativamente a JADE, dove un agente-proxy riceve le richieste dalla parte Web (servlet JSP), reperisce le informazioni necessarie comunicando con il sistema ad agenti ed infine restituisce i risultati alla servlet. In questo caso l'idea originale viene estesa, nell'ottica di ottenere una soluzione efficace ed affidabile per integrare una Web application ed un sistema multi-agente.

Le caratteristiche principali della soluzione proposta sono le seguenti:

- l'architettura ottenuta è facilmente integrabile con i sistemi multi-agente esistenti;
- l'inserimento di un layer tra la Web application ed il sistema multi-agente semplifica notevolmente il processo di sviluppo: sia i programmatori della parte Web sia quelli della parte ad agenti fanno solo riferimento al Web Agent, quindi lo sviluppo dei due sottosistemi è sostanzialmente indipendente;
- il Web Agent è in grado di filtrare i messaggi ricevuti nell'ottica di eliminare eventuali richieste duplicate (causate ad esempio dal refresh della pagina Web da parte dell'utente);
- il Web Agent può essere configurato per rispondere istantaneamente all'utente per far sì che l'utente stesso sia sicuro che la richiesta sia stata inviata correttamente e sia in fase di elaborazione da parte del sistema multi-agente; il problema della sincronizzazione utente-agente verrà discusso nella sezione successiva.

1.3.2 Sincronizzazione della Comunicazione fra Utente ed Agente

L'architettura descritta nella precedente sezione rappresenta, dal punto di vista dell'ingegneria del software, un'ottima integrazione fra tecnologia Web e tecnologia ad agenti, che consente di mantenere per i due sistemi due processi di sviluppo nettamente separati. Vi è tuttavia un problema ancora da analizzare dipendente dalle tecnologie utilizzate, che potrebbe causare un comportamento del sistema non conforme alle aspettative degli utenti e non ottimale in diverse situazioni. La causa di questo possibile problema è da imputarsi dalla mancanza di sincronizzazione fra le azioni degli utenti ed i comportamento del sistema multi-agente. In altre parole, si può verificare che la risposta fornita da un agente (che in generale ha la necessità di comunicare con un numero indefinito di altri agenti) può non essere tempestiva rispetto alla richiesta dell'utente, il quale, invece, interagendo con una Web application, si aspetta di osservare rapidamente i risultati delle proprie azioni. Inoltre, in diverse situazioni un agente potrebbe richiedere per primo di comunicare con un utente senza che questo abbia in precedenza effettuato alcuna azione.

Analizziamo ad esempio lo scenario in cui un utente delega all'agente ad esso associato (che successivamente verrà chiamato "Personal Agent") l'organizzazione di un meeting con altri utenti del sistema. L'agente inizia la negoziazione. Ovviamente questo processo occupa un tempo non definito a priori, non è istantaneo, e di conseguenza la Web application effettua il refresh della pagina Web non correttamente. Tuttavia, quando la negoziazione ha termine, per l'agente è impossibile forzare un refresh della pagina per far sì che l'utente visualizzi i risultati. Lo stesso problema si verifica ogni qual volta un utente interagisce con un agente. Dal momento che il problema è strettamente legato alle tecnologie impiegate, la soluzione risiede necessariamente in un miglioramento, in un'evoluzione delle tecnologie stesse.

Grazie all'avvento di tecnologie emergenti nel campo della programmazione Web, negli ultimi tempi è sempre più comune il concetto di applicazione Web

"desktop-like". In altri termini si sta assistendo all'avvento di nuovi paradigmi di programmazione che consentono di avere applicazioni ibride fra i classici programmi deskop e le classiche Web application. Questa tecnologia emergente è chiamata AJAX, acronimo di "Asynchronous JavaScript and XML", e consente di ottenere applicazioni Web interattive basate sullo scambio di piccole quantità di dati "on the fly" tra una parte client con cui si interfaccia l'utente e una parte server che rimane dietro le quinte, in modo che l'intera pagina Web non debba essere ricaricata ogni volta che l'utente interagisce con essa. È semplice comprendere come le caratteristiche di AJAX sposino perfettamente le necessità di sincronizzazione del canale di comunicazione agente-utente, rendendo possibile all'agente un pushing diretto delle informazioni all'interno della Web application e di conseguenza all'interno della pagina Web visualizzata dall'utente.

1.4 RAP - Remote Assistant for Programmers

RAP [35], acronimo di "Remote Assistant for Programmers", è il primo sistema implementato sulla base della piattaforma RAVE nato con l'obiettivo di dare supporto a gruppi di sviluppatori di codice Java fornendo loro pronte soluzioni ai problemi riscontrati nel corso dell'attività di programmazione.

1.4.1 Il Sistema RAP

Il reperimento di informazioni pertinenti rappresenta un problema di lunga data nel campo dell'informatica. Nel corso degli anni, approcci convenzionali come basi di dati, sistemi di recupero delle informazioni e motori di ricerca si sono occupati di questo problema ed in parte l'hanno risolto. Tuttavia accade spesso che le informazioni più preziose ed indispensabili non siano o non possano nemmeno essere catalogate od indicizzate all'interno di repository o database, ma possano essere reperite solamente effettuando una richiesta mirata ad una persona esperta nel particolare argomento. In questo senso spesso la possibilità di risolvere un particolare problema si riduce alla ricerca di una persona a cui sottoporre la

specifica richiesta di assistenza. È tuttavia impensabile che un esperto su un particolare topic sia sempre disponibile a rispondere in prima persona a richieste spesso ripetitive o banali, quindi l'utente che si trova in difficoltà, prima di interpellare l'esperto, dovrebbe avere la possibilità di consultare un repository di risposte già date e, solo nel caso non trovasse alcun documento utile, inviare specifica richiesta.

Sulla base di queste considerazioni, a partire dalla piattaforma RAVE descritta nelle precedenti sezioni, è stato sviluppato il sistema RAP, finalizzato al supporto di comunità di studenti e ricercatori impegnati su progetti basati sull'utilizzo del linguaggio di programmazione Java.

Il sistema RAP è stato sviluppato come parte del progetto @lis-TechNet [1]. Il collegamento con il progetto @lis-TechNet risulta evidente se si considera la struttura aperta e distribuita del sistema, che riflette l'architettura della rete sottostante atta a collegare diverse piattaforme ad agenti dislocate per il mondo in un'ottica prettamente peer-to-peer. Infatti l'obiettivo principale del progetto @lis-TechNet è proprio quello di "mostrare come la prossima generazione di tecnologie Web, tecnologie ad agenti e Web semantico possano essere combinate per creare applicazioni flessibili e dinamiche".

1.4.2 Gli Agenti del Sistema

Nel sistema RAP ad ogni utente viene associato un agente (detto in seguito "Personal Agent" o PA), che ha il compito di assisterlo nella risoluzione di problemi di programmazione, proponendogli, da un lato, informazioni estratte da repository e, dall'altro, inoltrandogli soluzioni e risposte fornite da altri utenti che presentano un profilo da "esperto" sul particolare topic. Un PA ha anche il compito di mantenere aggiornato il profilo dell'utente a lui collegato sia sulla base delle sue competenze estratte direttamente dal codice Java che l'utente produce, sia valutando di volta in volta le risposte e l'aiuto fornito ad altri utenti che lo interrogano su particolari problemi di programmazione.

Il sistema è basato su sette differenti tipologie di agenti: Personal Agent, Code Documentation Manager, Answer Manager, User Profile Manager, Mail Manager, Starter Agent e Directory Facilitator.

Gli agenti chiamati "Personal Agent" rappresentano una sorta di interfaccia fra gli utenti e le diverse parti del sistema ed ognuno di essi è responsabile della creazione e del mantenimento del profilo dell'utente associato quando l'utente è on-line. L'interazione fra utente e Personal Agent si svolge in due modalità: quando l'utente è attivo e collegato al sistema, attraverso una classica interfaccia Web-based, quando l'utente è off-line tramite e-mail. Solitamente il numero di Personal Agent attivi corrisponde al numero di utenti collegati al sistema, ma è possibile che un Personal Agent venga attivato per comunicare via e-mail con l'utente a lui associato che in quel momento si trova off-line.

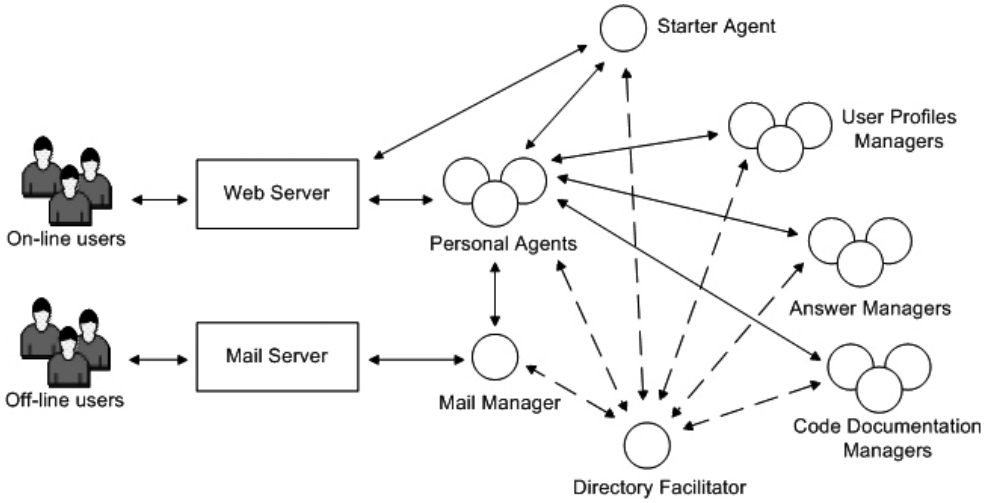

Figura 1.2 - Architettura del sistema RAP

Gli agenti chiamati "User Profile Manager" sono preposti alla gestione ed all'aggiornamento dei profili degli utenti off-line ed hanno l'importante compito di attivare un particolare Personal Agent nel momento in cui è necessario inviare una notifica via e-mail ad un utente del sistema in quel momento off-line.

Gli agenti chiamati “Code Documentation Manager” hanno un ruolo fondamentale in quanto sono i gestori della documentazione presente nel sistema ed hanno il compito di reperire, attingendo ai diversi repository, le informazioni corrette per soddisfare le richieste provenienti dagli utenti.

Gli agenti chiamati “Answer Manager” hanno una funzione simile ai Code Documentation Manager con la differenza che questi si occupano di aggiornare il repository delle risposte fornite dagli utenti del sistema e di valutare, in base alle nuove richieste di assistenza, se esistono risposte a problemi simili da poter riutilizzare. Oltre a fornire pronte soluzioni agli utenti in difficoltà, questi agenti hanno anche il compito di ricevere le valutazioni delle risposte da parte degli utenti che avevano richiesto assistenza e di inoltrarle sia ai Personal Agent sia agli User Profile Manager per far sì che i profili degli utenti autori delle risposte vengano aggiornati.

L’agente chiamato “Mail Manager” si occupa essenzialmente di gestire la comunicazione e-mail fra gli utenti off-line ed i relativi Personal Agent.

L’agente chiamato “Starter Agent” si occupa di attivare un Personal Agent nel momento in cui l’utente collegato ad esso diventa attivo nel sistema e ne richiede l’assistenza.

Infine l’agente chiamato “Directory Facilitator” è un agente standard della piattaforma JADE che ha il compito di comunicare ad un agente che lo richiede l’indirizzo di un agente attivo nel sistema che fornisce un determinato servizio, una sorta di servizio di “pagine gialle”.

La figura 1.2 mostra una schematizzazione di una piattaforma RAP sottolineando le interazioni fra le diverse tipologie di agenti. È interessante sottolineare che una piattaforma RAP può essere distribuita su diversi nodi computazionali e che un sistema RAP può essere composto da diverse piattaforme connesse via Internet l’una all’altra. Inoltre, in figura 1.2 le immagini che mostrano la sovrapposizione di più di un utente/agente suggeriscono che nel sistema può essere presente contemporaneamente più di un utente/agente di quel tipo. Infine, in un sistema RAP abbiamo un agente Directory Facilitator per ciascuna piattaforma.

1.4.3 Funzionamento del Sistema

Per descrivere il funzionamento del sistema RAP ipotizziamo uno scenario in cui uno degli utenti-programmatori del sistema chiede assistenza al proprio Personal Agent per risolvere un problema legato al codice Java che sta scrivendo ed il Personal Agent gli viene in aiuto fornendogli suggerimenti e risposte di diversi tipi. La descrizione dello scenario può essere suddivisa nei seguenti step:

1) selezione della tipologia di risposta/suggerimento da ricevere;
2) invio della richiesta di assistenza;
3) reperimento della risposta o del suggerimento adatto;
4) valutazione del suggerimento ricevuto.

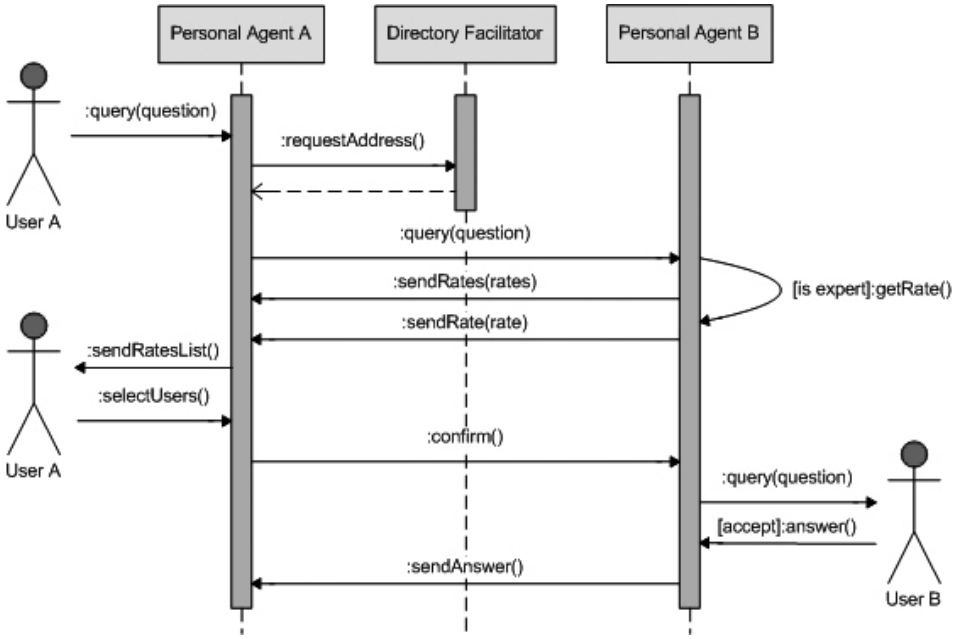

Figura 1.3 - Interaction diagram che descrive il funzionamento di RAP

Selezione della tipologia di risposta/suggerimento da ricevere: l'utente in difficoltà, in risposta alla propria richiesta, può ricevere informazioni estratte dal repository di codice del sistema, risposte selezionate dal repository di risposte già

fornite in passato o nuovi suggerimenti o risposte fornite dagli utenti attivi in quel momento. Prima di sottomettere la richiesta di assistenza, l'utente ha la possibilità di scegliere le tipologie di risposte che desidera ricevere dal proprio Personal Agent.

Invio della richiesta di assistenza: l'utente, attraverso l'interfaccia Web messa a disposizione dal sistema RAP, invia la richiesta di assistenza al proprio Personal Agent. In termini di programmazione Java, l'utente può chiedere aiuto su una specifica classe o su una serie di classi che sta utilizzando oppure richiedere assistenza su un problema legato alla particolare implementazione che sta affrontando. La richiesta (in gergo detta "query") è composta da due parti distinte. La prima parte, detta "annotation", identifica il contesto della richiesta: essa può contenere diverse keyword anche selezionabili da un glossario fornito dal sistema e/o il nome della classe sulla quale l'utente ha bisogno di aiuto e/o i nomi dei metodi che l'utente sta utilizzando. La seconda parte della richiesta è testuale ed a completa discrezione dell'utente.

Reperimento della risposta o del suggerimento adatto: il Personal Agent esegue diverse azioni interagendo con gli altri agenti del sistema per reperire tutte le informazioni necessarie a risolvere il problema dell'utente. Per il ritrovamento di informazioni disponibili nei repository di codice del sistema, il Personal Agent richiede al Directory Facilitator gli indirizzi di tutti i Document Manager attivi nel sistema ed invia loro la richiesta di assistenza. Questi valutano il problema e vanno alla ricerca di segmenti di codice che possano soddisfare la richiesta. Successivamente, per ricevere informazioni derivanti dai repository di risposte già date, il Personal Agent richiede al Directory Facilitator gli indirizzi degli Answer Manager disponibili che, una volta interrogati, rispondono al Personal Agent inviandogli risposte e suggerimenti associando ad ognuno di essi un punteggio che ne mostra la pertinenza rispetto alla richiesta ricevuta. Il reperimento di informazioni derivanti da nuove risposte fornite dagli utenti del sistema rappresenta un processo più complesso, che possiamo suddividere nei seguenti step:

3.1) ricerca degli utenti esperti sul particolare argomento;

3.2) valutazione degli utenti trovati;

3.3) selezione degli esperti ai quali chiedere aiuto;

3.4) ricezione delle risposte.

Ricerca degli utenti esperti sul particolare argomento: il Personal Agent interroga il Directory Facilitator per ricevere la lista degli altri Personal Agent attivi (associati quindi ad utenti on-line in quel momento) e la lista degli User Profile Manager del sistema (che, come detto in precedenza, sono responsabili della gestione dei profili degli utenti off-line). Ricevuti gli indirizzi degli agenti richiesti, il Personal Agent inoltra loro la richiesta di assistenza.

Valutazione degli utenti trovati: i Personal Agent e gli User Profile Manager interrogati, sulla base della richiesta ricevuta, assegnano ad ogni profilo utente una valutazione. A questo punto, gli agenti che dai calcoli effettuati ottengono punteggi soddisfacenti rispondono al Personal Agent che li aveva interrogati inviandogli la singola valutazione ottenuta (nel caso di un Personal Agent, ossia nel caso di un profilo di un utente on-line) o con un certo numero di valutazioni (nel caso di uno User Profile Manager).

Selezione degli esperti ai quali chiedere aiuto: ricevuta la lista degli esperti sull'argomento, il Personal Agent divide quelli on-line da quelli off-line, ordina entrambi gli elenchi sulla base della valutazione di ciascun utente ed infine li presenta all'utente. A questo punto sta all'utente scegliere uno o più utenti a cui chiedere effettivamente aiuto e, sulla base di questa scelta, il Personal Agent invia la query ai relativi Personal Agent (per gli utenti on-line selezionati) o agli User Profile Manager (per gli utenti off-line selezionati).

Ricezione delle risposte: i Personal Agent degli utenti on-line scelti dall'utente in difficoltà inoltrano immediatamente la richiesta di assistenza ai propri utenti e rispondono non appena gli utenti stessi prestano la propria assistenza; gli User Profile Manager invece, con la collaborazione dello Starter Agent, attivano i Personal Agent degli utenti off-line selezionati. Questi inoltrano la richiesta di assistenza ai propri utenti, al momento off-line, tramite e-mail e successivamente ritornano nello stato inattivo. Gli utenti off-line possono prestare la propria assistenza rispondendo al messaggio e-mail o loggandosi al sistema ed utilizzando l'interfaccia Web di RAP. In caso di risposta via e-mail da parte di un utente, sarà

compito del Mail Manager attivare il relativo Personal Agent che si occuperà di estrarre la risposta dal messaggio e-mail e di inoltrarla al Personal Agent dell'utente che aveva richiesto assistenza. Ogni volta che il Personal Agent riceve una risposta, la presenta immediatamente al proprio utente.

Dopo aver ricevuto le risposte da tutti gli esperti selezionati o dopo che è stata raggiunta una deadline temporale preimpostata o, ancora, nel momento in cui l'utente risolve il proprio problema grazie ed uno dei suggerimenti ricevuti, il compito dell'utente consiste nella valutazione delle risposte ricevute affinché il sistema possa essere aggiornato. Questo significa che il Personal Agent associato all'utente si occupa di inoltrare la valutazione ricevuta relativamente ad ogni risposta agli altri agenti del sistema che aggiornano sia il profilo dell'utente che ha fornito quella risposta sia la valutazione della risposta stessa all'interno dei repository. È interessante sottolineare che la valutazione di una determinata risposta non viene inoltrata all'utente autore della risposta stessa (il meccanismo della valutazione serve solamente al sistema per evolversi), inoltre agli utenti interrogati che non forniscono alcun suggerimento viene automaticamente attribuita una valutazione negativa. Questo è inevitabile dal momento che RAP si basa essenzialmente sull'aspetto collaborativo della sua community di utenti: nel momento in cui un utente non partecipa vengono a cadere le basi concettuali del sistema e l'utente diventa virtualmente inutile.

1.4.4 Gestione dei Profili Utente e della Documentazione

Nel sistema RAP, la gestione dei profili degli utenti e della documentazione si struttura in due fasi distinte: una fase di inizializzazione ed una fase di aggiornamento.

Per rendere il sistema più snello e veloce possibile e per evitare di avere profili incompleti od informazioni non del tutto corrette, la fase di inizializzazione in cui vengono creati i profili degli utenti e dei documenti è stata automatizzata affidandosi ad un algoritmo molto simile a quello integrato nel sistema Expert

Finder [3]. I profili vengono rappresentati da vettori associati a determinate parole chiave ed il modulo di ciascun vettore è legato, per il profilo di un documento, a quanto frequente è la keyword all'interno del documento stesso o, per il profilo di un utente, a quanto spesso l'utente utilizza quella keyword nel proprio codice Java. Il set di keyword utilizzato per creare i vari profili, compresi quindi i nomi delle classi e dei metodi delle librerie Java utilizzate dalla community, va a costituire il glossario di sistema messo poi a disposizione degli utenti per creare le proprie richieste di assistenza.

L'algoritmo usato per la costruzione dei profili degli utenti e dei documenti è detto TF-IDF (Term Frequency Inverse Document Frequency) [19]. L'applicazione di questo algoritmo ha causato alcuni problemi legati alla natura distribuita e multi-piattaforma del sistema RAP: questi problemi e le relative soluzioni saranno discussi nella sezione successiva.

I profili degli utenti sono inizialmente generati dai relativi Personal Agent attraverso l'analisi del loro codice Java. Ovviamente tali profili vengono poi aggiornati e si evolvono nel tempo sulla base del nuovo codice prodotto e dal comportamento degli utenti stessi in relazione alle richieste di assistenza che richiedono e che forniscono. L'aggiornamento di un profilo di un utente o di un documento viene effettuato nelle tre seguenti situazioni: i) un utente richiede assistenza ed associa una valutazione ad ognuna delle risposte ricevute, ii) viene introdotta una nuova libreria Java fra quelle utilizzate dalla community oppure viene ampliato il glossario di termini del sistema, e iii) un utente scrive nuovo codice Java.

Nel primo caso è possibile distinguere tre possibilità a seconda di qual'è la sorgente del suggerimento (un utente, il repository di documenti o il repository di risposte già date). Se il suggerimento proviene da un utente, sulla base della valutazione associata alla risposta il sistema aggiorna il profilo dell'utente relativamente alle keyword presenti nella parte di annotation della query. Inoltre, se la valutazione è positiva, la risposta viene inclusa all'interno del repository delle risposte e viene generato un profilo legato ad essa sulla base della parte di annotation della query e della valutazione ricevuta. Se la risposta proviene dal

repository dei documenti e la valutazione è positiva, essa viene aggiunta al repository delle risposte ed il relativo profilo viene generato sulla base del profilo del documento originale e della valutazione ricevuta. Se invece la risposta proviene dal repository delle risposte e la valutazione è positiva, viene aggiornata la parte di profilo della risposta associata ai termini presenti nella parte di annotation della query sulla base della valutazione ottenuta. Inoltre, nel caso in cui questa risposta valutata positivamente sia stata fornita in passato da un utente (e non derivi invece dal repository dei documenti), viene aggiornato anche il profilo dell'utente autore sempre sulla base dei termini presenti nella parte di annotation della query. Infine all'interno del sistema viene tenuta traccia della query legandola alla risposta che ha ricevuto valutazione positiva.

È possibile notare che non si fa mai riferimento al caso in cui un suggerimento ottenga una valutazione negativa, questo perché nella fase di progettazione del sistema RAP si è deciso di ignorare tutte le valutazioni negative. Il ragionamento alla base è molto semplice: se una risposta è presente nel repository delle risposte del sistema, significa che in passato almeno un utente ha considerato questa risposta utile nella risoluzione di un proprio problema valutandola positivamente. Se, successivamente, questa risposta riceve una valutazione negativa, questo significa che nel caso specifico essa non ha portato alla soluzione del problema, ma rimane comunque il fatto che in passato la risposta almeno una valutazione positiva l'aveva ricevuta, quindi è ragionevole che venga mantenuta all'interno del relativo repository.

Quando viene introdotta una nuova libreria software Java fra quelle utilizzate dalla community di utenti di RAP, oppure quando viene ampliato il glossario di sistema tramite l'inserimento di nuovi termini o keyword, è necessario che tutti i profili dei documenti e degli utenti vengano aggiornati. Mentre i profili dei documenti vengono rigenerati sulla base del nuovo set di termini disponibili, i profili degli utenti vengono aggiornati con l'inserimento di nuovi vettori associati ai nuovi termini i cui moduli saranno legati, secondo l'algoritmo TF-IDF, alla frequenza dei termini stessi all'interno del codice scritto dagli utenti.

Infine, il profilo degli utenti viene aggiornato con l'aggiunta di nuovi termini pesati (ossia nuovi vettori), ogni qual volta essi generano nuovi segmenti di codice Java.

1.4.5 Comunità Distribuite ed Aperte

Un importante obiettivo che ha guidato la progettazione di RAP è stato senza dubbio la creazione di un sistema che fornisse supporto a comunità di utenti aperte e distribuite. Infatti, la ricerca di esperti e di informazioni diventa molto agevole e trae sicuramente grandi benefici se la comunità alla base del sistema ha la capacità di crescere e di includere i nuovi utenti o nuove comunità.

Questo rappresenta un aspetto fondamentale del sistema, dal momento che RAP è stato inizialmente sviluppato per una comunità dinamica ed aperta di utenti connessi alla rete openNet di piattaforme ad agenti. Infatti la prima versione del sistema è stata sviluppata all'interno del progetto europeo @lis-TechNet e l'obiettivo principale di questo progetto consiste proprio nello sviluppo di reti aperte basate su tecnologia ad agenti.

Dalle considerazioni fatte nelle precedenti sezioni, è facile comprendere come RAP si basi su una struttura completamente aperta, non solo perché il sistema supporta la continua registrazione e quindi l'accesso di nuovi utenti (ed ovviamente l'eliminazione di utenti registrati), ma anche perché gli utenti facenti parte del sistema possono far sì che i propri profili si evolvano tramite l'acquisizione di nuove skill o la scrittura di nuovo codice Java.

Inoltre, la comunità alla base di RAP rappresenta una community distribuita dal momento che il sistema globale può essere costituito da una serie di sottosistemi e, di conseguenza, di comunità locali. Ciascuna di queste può esistere ed operare in modo isolato, ma può anche decidere di prendere parte ad un gruppo di comunità, condividendo con gli altri membri del gruppo i propri esperti ed i propri repository di informazioni. La costituzione ed il disassemblamento di gruppi di comunità sono operazioni del tutto dinamiche in quanto, come detto in precedenza, le singole community sono completamente indipendenti, in modo simile ai componenti di una rete peer-to-peer.

La natura aperta e distribuita del sistema RAP fornisce le migliori condizioni per il recupero e la condivisione di informazioni, ma comporta anche alcuni problemi non irrilevanti nella valutazione e nella gestione di tali informazioni. Infatti, la valutazione di informazioni e di esperti su un particolare argomento è fortemente legata all'effettiva composizione della community. Ad esempio, se un utente viene individuato come massimo esperto nella soluzione di un particolare problema, questa valutazione dipende essenzialmente dall'analisi dei profili degli utenti in quel momento registrati al sistema. Se successivamente il sistema viene ampliato dall'ingresso di una nuova comunità di utenti, è possibile che l'utente precedentemente individuato venga surclassato da un nuovo utente più esperto. In questo caso, scendendo nello specifico, un'informazione come "l'utente A ha utilizzato n volte un metodo della classe X" è ancora valida, ma un'informazione come "l'utente A è il massimo esperto nell'utilizzo della classe X" può cambiare in relazione alla nuova composizione della comunità globale di utenti.

Il problema deriva dal fatto che, mentre le informazioni legate al profilo personale di ogni utente sono ancora valide, il rating e tutte le altre informazioni relative alla comunità devono essere ricalcolate. L'algoritmo TF-IDF è notoriamente utilizzabile in modo molto semplice ed agevole all'interno di sistemi nei quali la gestione dei profili e delle informazioni avviene in modo centralizzato. Nel caso di RAP la questione si complica: il sistema è distribuito, solamente i Personal Agent possono accedere al codice scritto dai relativi utenti (per questioni di privacy e di sicurezza) e l'aggiornamento dei profili degli utenti viene gestito sia dai Personal Agent, sia dagli User Profile Manager nel momento in cui il Personal Agent associato all'utente il cui profilo deve essere aggiornato non è attivo.

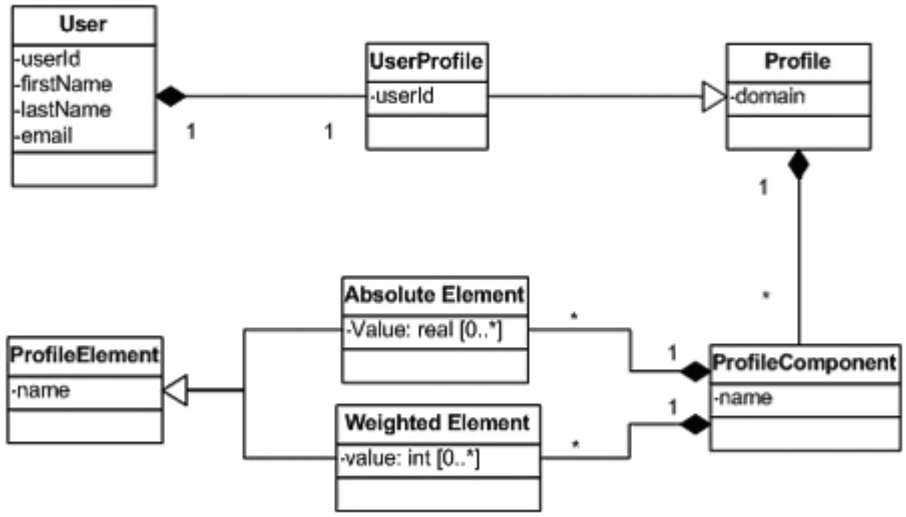

Figura 1.4 - Class diagram per i profili utente

Per queste ragioni, ogni profilo in RAP è composto da due elementi: un elemento di valenza assoluta (Absolute Element) ed un elemento relativo (Weighet Element) relazionato alla composizione del sistema. Il primo dipende solo dal profilo dell'utente (o del documento) e viene descritto secondo le modalità esposte nella precedente sezione. Il secondo elemento dipende sia dal profilo dell'utente, sia dagli altri profili presenti nella community. La figura 1.4 mostra la struttura del class diagram UML relativamente ai profili utente.

Le situazioni in cui può essere necessario un aggiornamento dell'elemento relativo del profilo di uno o più utenti sono abbastanza frequenti:

1) una nuova comunità di utenti entra a far parte del sistema o una comunità abbandona il sistema;
2) un nuovo utente entra a far parte della comunità o un utente abbandona la comunità;
3) cambiano alcune parti di un profilo utente: ad esempio l'utente scrive del nuovo codice Java oppure fornisce un suggerimento che riceve una determinata valutazione.

Per motivi di performance, in particolare se la comunità di utenti è molto numerosa, il processo di aggiornamento potrebbe essere troppo costoso in termini di risorse di sistema. Per questa ragione l'amministratore di RAP ha la facoltà di decidere che il sistema effettui tutte le operazioni di aggiornamento e ricalcoli i profili utente solo in determinati momenti.

1.4.6 Sistemi per la Ricerca di Informazioni ed Esperti

Negli ultimi anni i sistemi per la ricerca di informazioni hanno riscosso un crescente interesse, tant'è che è possibile rintracciare facilmente diversi progetti volti allo sviluppo di tool di e-learning o di applicazioni di supporto alla programmazione. Alcuni dei più famosi ed importanti sistemi sono però stati studiati per la ricerca di pagine Web e, pur supportando un certo livello di personalizzazione in base all'utilizzo che se ne fa, essi non sono di grande aiuto per comunità che vedono nella programmazione la propria attività principale.

GroupLens [42] rappresenta il primo sistema ad avere adottato un approccio collaborativo nella ricerca di informazioni. Questo sistema cerca di individuare analogie fra i profili degli utenti della comunità e, quando un utente ricerca una determinata informazione, risponde all'utente consigliandoli la consultazione di documenti che erano stati valutati positivamente da utenti a lui "simili". Adaptive Web Site Agent [37] è invece un sistema ad agenti che lavora sulla documentazione di un sito Web e consiglia agli utenti del sito determinati documenti sulla base di diversi criteri quali le preferenze dell'utente sul particolare argomento o la frequenza di apparizione nei documenti di particolari keyword ricercate dall'utente.

Altri tipi di sistemi supportano anche la ricerca di utenti esperti sul particolare argomento. Ad esempio il sistema Expert Finder [3] presenta diverse analogie con RAP, nel senso che è in grado di ricercare, relativamente ad un particolare problema di programmazione Java, utenti esperti a cui chiedere assistenza. Questo sistema analizza il codice Java scritto dalla comunità e crea i profili degli utenti sulla base di un insieme di caratteristiche rilevanti del linguaggio Java confrontate con le classi scritte dagli utenti stessi. I profili così creati vengono poi utilizzati per

ricercare esperti nel momento in cui un utente incontra un problema di programmazione e richiede assistenza. MARS [10] è invece un sistema di supporto basato sul concetto di "social network". Questo sistema è completamente distribuito e comprende agenti che preservano la privacy e l'autonomia dei vari utenti. Questi agenti costituiscono una social network legandosi l'uno all'altro sulla base delle proprie competenze nella soluzione di problemi in determinati campi e, grazie a questa rete, hanno la possibilità di ricercare agevolmente esperti per fornire assistenza agli utenti in difficoltà.

Altri interessanti progetti riguardano invece l'implementazione di sistemi per il supporto di attività di e-learning, quali WBT (Web Based Teaching) [46], un sistema multi-agente al supporto di attività di insegnamento e di apprendimento basato su Web services. Il sistema consiste in un sito Web contenente materiale didattico (il tema è sempre quello della programmazione) ed in una sorta di bacheca elettronica tramite la quale gli utenti del sistema possono accedere a suggerimenti per risolvere i problemi incontrati durante la propria attività di programmazione. In questo sistema gli agenti hanno il compito di gestire le richieste di assistenza richiedendo la consulenza degli utenti-docenti o di altri utenti-studenti che in precedenza avevano affrontato il medesimo problema. Un altro progetto degno di interesse è I-MINDS [50], incentrato su un sistema di e-learning che consente ad utenti-studenti sia di prendere parte in modo attivo a gruppi di studio virtuale, sia di fruire in modo passivo di contenuti didattici multimediali. Questo sistema è basato su tre differenti tipologie di agenti: agenti-docenti, agenti-studenti ed agenti-proxy. Gli agenti-docenti interagiscono con gli agenti-studenti ed hanno il compito di: i) inviare informazioni didattiche agli agenti-studenti ed agli agenti-proxy, ii) gestire i profili degli utenti-studenti e, sulla base di questi, generare esercizi personalizzati, iii) gestire le domande inviate dagli utenti-studenti e iv) gestire la sessione del gruppo virtuale di studio o di insegnamento. Gli agenti-studenti hanno invece il compito di analizzare i profili degli utenti-studenti per identificare eventuali utenti utili a rispondere ad una particolare richiesta di assistenza. Gli agenti-proxy sono invece i gestori della comunicazione fra gli utenti-docenti e gli utenti-studenti ed assumono un ruolo

fondamentale nel momento in cui un utente-studente ha problemi di banda ed è necessario un livello avanzato di filtraggio dei messaggi per ridurre il traffico di rete. L'ultimo sistema analizzato in tema e-learning è chiamato Guardia Agent, un sistema ad agenti finalizzato al supporto di team di studenti impegnati su progetti comuni [28]. Questo sistema associa un agente a ciascuno studente, che in modo autonomo mantiene monitorata l'evoluzione complessiva del progetto, suggerisce le modalità con cui partecipare al miglioramento del progetto stesso e gestisce la comunicazione fra i membri del team.

RAP presenta molti aspetti in comune con i progetti descritti, in particolare con WBT, I-MINDS ed Expert Finder. Infatti, la caratteristica che accomuna questi sistemi e che rappresenta anche un punto di forza di RAP è la presenza di un sistema ad agenti finalizzato alla ricerca di utenti esperti in grado di fornire assistenza su particolari problemi. Tuttavia, a differenza di RAP, nessuno dei sistemi citati integra il reperimento di informazioni da diverse sorgenti (ricordiamo che in RAP la soluzione ad un problema può giungere da un esperto ma anche dal repository delle risposte o dal repository dei documenti) ed inoltre nessuno dei sistemi citati presenta una gestione dei profili utente simile a RAP, in cui i profili sono costantemente aggiornati considerando sia l'attività di programmazione degli utenti, sia il comportamento degli utenti stessi in relazione alle richieste di assistenza che ricevono da parte di utenti in difficoltà. Un altro contributo originale di RAP consiste nella creazione di un sistema di supporto composto da un gruppo aperto e distribuito di comunità di utenti, dove ogni comunità è completamente indipendente e può, in ogni momento ed in maniera del tutto dinamica, associarsi od abbandonare un gruppo.

1.5 CAFE - Collaborative Agents for Filtering E-mails

CAFE [31][32][34], acronimo di "Collaborative Agents for Filtering E-mails", rappresenta il secondo sistema multi-agente sviluppato sulla piattaforma RAVE e

può essere considerato, come RAP, un sistema di supporto ad una comunità di utenti. In questo caso però l'obiettivo non è quello di fornire assistenza alle attività della community, ma quello di risolvere, tramite un approccio collaborativo basato sulla tecnologia multi-agente, un problema che ormai affligge quotidianamente la quasi totalità degli utenti Internet: la ricezione di messaggi illegittimi, in gergo chiamati "spam".

1.5.1 Il Problema dello Spam

Negli ultimi anni, la diffusione delle tecnologie basate su Internet ha cambiato radicalmente il modo di comunicare: la posta elettronica è entrata a far parte della quotidianità e rappresenta al giorno d'oggi uno dei mezzi di comunicazione più utilizzati. Il successo della e-mail è ovviamente da imputare alle enormi potenzialità che questa tecnologia di scambio di informazioni mette a disposizione dei propri utenti: la possibilità di comunicare con diverse persone in modo semplice, rapido ed economico ha fatto della posta elettronica il canale globale di comunicazione per eccellenza anche a livello business.

Tuttavia, il sempre crescente utilizzo di questo potente mezzo ha fatto sì che il volume di e-mail ricevute giornalmente in media dagli utenti del Web crescesse esponenzialmente, considerando soprattutto il fatto che molti dei messaggi che ogni giorno si ricevono sono messaggi indesiderati, chiamati tecnicamente "spam". Per definizione, con il termine "spam" (o "junk mail" o "bulk mail") si fa riferimento a tutti quei messaggi (pubblicitari o non) inviati senza il permesso del destinatario. Lo "spamming" è un comportamento ampiamente considerato inaccettabile dagli ISP (Internet Service Provider) e dalla maggior parte degli utenti di Internet. Mentre questi ultimi trovano lo spam fastidioso e con contenuti spesso offensivi, gli ISP vi si oppongono per i costi del traffico generato dall'invio indiscriminato e per i non trascurabili problemi di sovraccarico di banda provocati.

Negli ultimi anni la gravità del problema ha raggiunto livelli allarmanti: il numero di e-mail di spam ricevute giornalmente dagli utenti del Web è in continuo aumento e questo comporta problemi sempre crescenti agli utenti sia a livello di tempo e di occupazione di banda (e quindi di costi di connessione), sia a livello di

contenuti ricevuti, spesso volgari ed offensivi. Una ricerca condotta nel 2004 dalla società Kaspersky Labs, azienda leader nello sviluppo di software antivirus, aveva stimato che la percentuale di e-mail di spam rispetto all'intero numero di e-mail scambiate sul Web fosse del 70%, mentre ricerche più recenti stimano tale percentuale addirittura a cifre superiori al 90%. Inoltre, relativamente all'utilizzo quotidiano della posta elettronica, lo spam non è l'unico problema da sottolineare: se un utente utilizza in modo massivo la propria casella e-mail e riceve quotidianamente un elevato numero di messaggi, è possibile che messaggi importanti od urgenti vengano ignorati: sarebbe in questo senso molto utile avere strumenti di classificazione automatica ed intelligente delle e-mail in base a criteri definiti dall'utente stesso.

La classificazione delle e-mail e l'individuazione dello spam rappresentano task molto difficili e delicati per diverse ragioni. Le e-mail giungono ovviamente da mittenti diversi e spesso non conosciuti e la rilevanza del messaggio non sempre può essere dedotta dall'oggetto del messaggio stesso. L'individuazione dello spam è ancora più complessa: lo spam è in continua evoluzione in termini di argomenti e gli spammer cercano in ogni modo di rendere i propri messaggi il più possibile simili a messaggi legittimi per ingannare i filtri antispam.

1.5.2 Analisi delle Tecniche di Classificazione

L'individuazione dello spam e la classificazione dei messaggi sulla base della loro rilevanza sono problemi oggetto di ricerche da diversi anni a questa parte. Sono stati proposti diverse soluzioni per risolvere il problema dello spamming, ma nessuna di esse può essere considerata completamente soddisfacente. Le metodologie proposte nel corso degli anni possono essere suddivise in due categorie: metodologie basate su un approccio "statico" e metodologie basate su un approccio "dinamico".

L'approccio statico si propone di identificare i messaggi di spam tramite il semplice confronto dell'indirizzo del mittente con una o più liste estratte da una serie di database DNSBL (DNS-based Black Lists) [11]. In questo senso diversi mail server mantengono un elenco di indirizzi di possibili spammer e segnalano

come spam ogni messaggio proveniente da essi. È facilmente intuibile che un approccio di questo tipo non possa garantire risultati soddisfacenti, dal momento che tali elenchi non vengono realizzati tenendo in considerazione i contenuti dei messaggi ed ogni e-mail viene valutata nello stesso modo, indipendentemente dalla sua struttura e, soprattutto, dal suo contenuto. La quasi totalità degli spammer sono quindi in grado di raggirare queste tipologie di filtri antispam semplicemente variando di continuo l'indirizzo e-mail dal quale i messaggi vengono inviati. È quindi possibile affermare che qualsiasi metodo di individuazione dello spam che si basi semplicemente su un approccio di tipo statico abbia una bassissima efficacia proprio perché in questo modo non si tiene conto della natura intrinsecamente dinamica del problema.

Altre metodologie più complesse sono basate su un approccio dinamico. Queste non si limitano a valutare la sorgente del messaggio ma effettuano un'analisi del contenuto del messaggio stesso e, sulla base di questa, utilizzando diverse tipologie di algoritmi, classificano il messaggio etichettandolo o meno come spam. La maggior parte di queste metodologie utilizza algoritmi di analisi e classificazione del testo basate sull'implementazione di tecniche di apprendimento automatico (note come "machine learning"). È possibile reperire diversi esempi di algoritmi di apprendimento applicati alla classificazione del testo (Lewis, 1992 [9]; Apte and Damerau et al., 1994 [4]; Dagan et al., 1997 [22]), che si propongono di suddividere documenti in categorie predefinite, dopo essere stati addestrati su un set preliminare di documenti noti. Algoritmi di questo tipo sono anche stati utilizzati nella classificazione delle e-mail (Cohen, 1996 [49]).

Uno dei primi tentativi di applicazione di tecniche dinamiche al problema dello spam è da imputare a Sahami et al. [38], che, tramite l'utilizzo di un classificatore Naive Bayes (uno dei più famosi metodi di apprendimento bayesiano), si propongono di distinguere i messaggi illegittimi da quelli legittimi, raggiungendo un buon livello di precisione. Di primo acchito potrebbe essere sorprendente che tecniche di classificazione del testo si rivelino efficaci nel filtraggio dello spam: infatti in generale un messaggio di spam è tale in quanto è inviato in modo indiscriminato ad un numero elevato e spesso casuale di destinatari, non per il

contenuto del messaggio stesso. Tuttavia è anche possibile notare che molte volte il contenuto dei messaggi di spam si ripete e fa spesso riferimento ad argomenti difficilmente citati all'interno di messaggi legittimi: proprio per questo motivo l'utilizzo di algoritmi di apprendimento può rivelarsi una scelta vincente.

Gli algoritmi bayesiani sono stati frequentemente usati nella progettazione di sistemi intelligenti per l'individuazione dei messaggi e-mail illegittimi (Androutsopoulos et al., 2000 [21]; McCallum and Nigam, 1998 [2]; Sanchez et al., 2002 [44]); è tuttavia possibile individuare alcune limitazioni anche in questa tipologia di approccio, legate soprattutto all'inefficacia riscontrata nel filtraggio di messaggi che non obbediscono, dal punto di vista lessicale e strutturale, ai canoni classici dello spam. Ad esempio, i filtri bayesiani incontrano molti problemi nell'analisi di messaggi contenenti molte immagini e poco testo (magari un semplice link). Un altro problema caratteristico delle tecniche di filtraggio basate su algoritmi bayesiani è l'elevato intervallo temporale richiesto per la fase di addestramento dei filtri stessi: è infatti necessario un periodo iniziale durante il quale il sistema analizzi un certo numero di messaggi per creare un proprio vocabolario di termini. Infine, i filtri bayesiani possono essere agevolmente ingannati inserendo all'interno dei messaggi in modo casuale una serie di termini conosciuti e leciti per far sì che il testo del messaggio sia considerato complessivamente legittimo.

Un approccio leggermente alternativo al problema, adottato da Jung et al. [27], consiste nell'implementazione di un sistema multi-agente che si occupa di filtrare i messaggi e-mail ricevuti dagli utenti del sistema stesso tramite l'analisi di alcune caratteristiche predefinite dei messaggi (ad es. l'oggetto) ed una serie di frasi-chiave estratte dall'oggetto e dal corpo dei messaggi stessi.

Alternativamente all'approccio descritto basato sull'analisi del contenuto testuale dei messaggi, recentemente sono state concepite nuove tecniche basate su un approccio collaborativo che non prende in considerazione il contenuto delle e-mail ma si basa sulla collaborazione di gruppi di utenti che condividono informazioni sullo spam ricevuto. La logica alla base di queste tecniche è molto semplice: quando viene ricevuto un messaggio di spam, l'utente che lo riceve crea

una sorta di firma del messaggio servendosi di un particolare algoritmo e condivide questa firma con il resto della comunità. Se successivamente lo stesso messaggio viene inviato ad un altro utente della comunità, esso viene immediatamente eliminato dal filtro dell'utente in quanto la firma di questo messaggio si trova già nello spam repository del sistema. Questo approccio richiede due elementi fondamentali: un efficace algoritmo di firma dei messaggi ed un sistema che consenta di condividere agevolmente e rapidamente le firme. La condivisione delle firme può essere centralizzata tramite meccanismi di clearing-house o, ancora meglio, distribuita tramite tecniche peer-to-peer, rendendo in questo modo il sistema molto più solido. Il sistema più famoso in questo ambito si chiama Vipul's Razor [47], anche conosciuto come SpamNet. Questo sistema si basa su una clearing-house centralizzata per la condivisione delle firme dei messaggi di spam ed un algoritmo di firma molto sofisticato.

Il problema della classificazione delle e-mail sulla base della loro rilevanza si collega all'associazione di un livello di reputazione agli utenti di un sistema o agli agenti di una comunità, che rappresenta un importante argomento di ricerca relativo ai sistemi multi-agente. Hertzum et al. [36] risolvono il problema assegnando un livello di fiducia agli utenti calcolato sulla base di informazioni estratte da diverse sorgenti. Nell'articolo è posta particolare enfasi sulla natura collaborativa di tale estrazione di informazioni. In particolare, viene presentato il sistema COGITO il cui obiettivo è quello di specificare il comportamento e le funzionalità di un sistema ad agenti al supporto di utenti durante la ricerca di informazioni e la presa di decisioni all'interno di siti Web di e-commerce.

Golbeck et al. [24] descrivono un algoritmo in grado di generare livelli di reputazione calcolati localmente per gli utenti di una semantic Web social network. Un'applicazione dell'algoritmo è TrustMail, un client e-mail che, sulla base del livello di reputazione del mittente di ciascun messaggio, è in grado di suddividere la posta in arrivo in termini di importanza. Questo progetto non si basa sulla tecnologia ad agenti, tuttavia fornisce un esempio di approccio collaborativo al problema della classificazione della posta sulla base di un livello di fiducia associato a ciascun utente della rete.

Lo sviluppo del sistema CAFE parte proprio dalle considerazioni fatte fino ad ora sui sistemi e sulle tecniche esistenti: l'intento, come vedremo, è quello di creare un sistema che, combinando un approccio collaborativo al problema e diverse metodologie di analisi dei messaggi, sia in grado di ridurre il numero di messaggi illegittimi ricevuti dalla sua comunità di utenti.

1.5.3 Il Sistema CAFE

CAFE, acronimo di "Collaborative Agents for Filtering E-mails", è un sistema multi-agente studiato per affrontare il problema dello spam tramite un approccio collaborativo e fare in modo che gli utenti del sistema stesso ricevano il minor numero possibile di messaggi illegittimi. In CAFE ad ogni utente viene associato un agente chiamato "E-mail Proxy Agent" (EPA), che rappresenta una sorta di interfaccia fra il client e-mail utilizzato dall'utente (Microsoft Outlook, Thunderbird, ecc.) ed il server di posta del sistema. Questa tipologia di agenti ha un ruolo fondamentale all'interno del sistema in quanto è responsabile della classificazione finale di ogni messaggio scaricato dal server e-mail in una delle tre categorie possibili: messaggi legittimi (detti anche "ham"), messaggi di spam o presunti messaggi di spam (che chiameremo "spam-presumed"). I messaggi legittimi sono a loro volta suddivisi a seconda che giungano da un mittente conosciuto, da un mittente totalmente sconosciuto, o da un mittente conosciuto da un altro utente del sistema (v. figura 1.5). Nel caso il messaggio giunga da un mittente non direttamente conosciuto dall'utente destinatario ma conosciuto da almeno uno degli altri utenti del sistema, il sistema assegna al messaggio un punteggio sulla base della reputazione assegnata al mittente dagli altri utenti del sistema che lo conoscono (il punteggio assegnato al messaggio va da 1 a 10 e rappresenta la media delle valutazioni assegnate al mittente del messaggio).

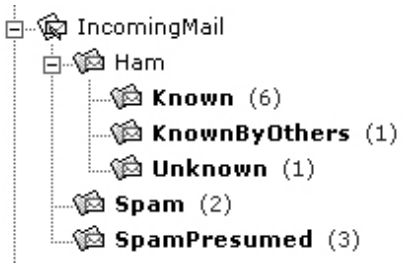

Figura 1.5 - Esempio di classificazione dei messaggi da parte di CAFE

Per effettuare la classificazione delle e-mail, i vari E-mail Proxy Agent sono supportati da altre tipologie di agenti che si occupano dell'analisi specifica dei vari messaggi. L'individuazione dello spam in CAFE si basa su tre tipologie distinte di analisi: un primo approccio basato sull'utilizzo di una funzione di hash, un approccio statico basato sull'uso di database DNSBL ed un approccio dinamico basato su filtro bayesiano. L'efficacia del sistema, per la natura del sistema stesso, è strettamente legata al comportamento degli utenti. Come vedremo nelle prossime sezioni, agli utenti di CAFE sono richieste due operazioni fondamentali: la segnalazione dei messaggi di spam ricevuti e la valutazione dei propri contatti.

1.5.4 Gli Agenti del Sistema

CAFE si basa su otto differenti tipologie di agenti: E-mail Proxy Agent, Digest Manager, Analysis Manager, DNSBL Agent, Bayesian Filter Agent, User Profile Manager, Starter Agent e Directory Facilitator.

Gli agenti chiamati "E-mail Proxy Agent", di cui si è già parlato nella sezione precedente, sono una sorta di interfaccia fra gli utenti ed il sistema ad agenti e gestiscono il flusso di e-mail dal server ai vari client. Un EPA viene attivato nel momento il cui l'utente associato ad esso diventa attivo nel sistema avviando il proprio programma di posta elettronica. Mentre in un sistema tradizionale il client e-mail si connette direttamente al server di posta ed effettua il download dei nuovi messaggi, in CAFE questa connessione diretta non avviene ma i nuovi messaggi vengono scaricati dall'agente EPA che, dopo averli analizzati e classificati, li invia al client. Come detto in precedenza, nella fase di analisi e di classificazione dei

messaggi gli agenti EPA sono supportati dalle altre tipologie di agenti del sistema. In particolare, essendo l'analisi effettuata a vari livelli, si richiede l'intervento dei seguenti agenti: Digest Manager, Analysis Manager, Bayesian Filter Agent, DNSBL Agents e User Profiles Managers.

Gli agenti chiamati "Digest Manager" sono responsabili della prima fase di analisi che, tramite un approccio basato sul confronto del digest di ogni messaggio con un database di digest di messaggi già riconosciuti come spam, rappresenta il primo tentativo del sistema di individuare messaggi illegittimi. Il digest di un messaggio e-mail è una sorta di firma del messaggio stesso ottenuta in modo univoco utilizzando una particolare funzione chiamata "one-way hash function".

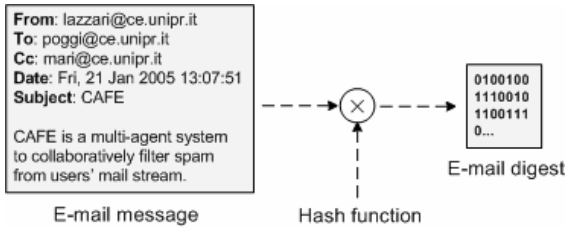

Figura 1.6 - Generazione del digest di un messaggio

La funzione di hash utilizzata in CAFE è un'implementazione Java della nota funzione MD5 [39]. Se l'agente Digest Manager riscontra una corrispondenza fra il digest del messaggio ed uno dei digest presenti nel database, etichetta immediatamente il messaggio come spam e la fase di analisi ha termine.

Gli agenti chiamati "Analysis Manager" si occupano invece della seconda fase di analisi, alla quale un messaggio viene sottoposto solo nel caso in cui non sia già stato catalogato come spam nella prima fase. La seconda fase utilizza un approccio più classico basato su un'analisi sia statica che dinamica del messaggio (v. sezione 1.5.2). Gli agenti Analysis Manager ricevono i messaggi da analizzare dagli E-mail Proxy Agent e li inoltrano agli altri agenti (descritti in seguito) che si occupano dell'analisi statica e dell'analisi dinamica basata su algoritmi bayesiani. In relazione ai risultati ottenuti nelle due tipologie di analisi, gli agenti Analysis

Manager classificano i messaggi come ham (messaggi legittimi) o come spam-presumed ed informano di conseguenza gli agenti EPA.

Gli agenti chiamati "DNSBL Agent" si occupano dell'analisi statica dei messaggi. Come è possibile intuire dal nome stesso di questa tipologia di agenti, l'approccio statico si basa sul confronto del mittente di ogni messaggio con le liste di IP e di domini bannati ottenuti dai database DNS-based Black Lists [11]. Dopo quest'analisi statica, ad ogni messaggio viene assegnata una valutazione (chiamata "static score") e tale valutazione viene comunicata agli agenti Analysis Manager. Ovviamente la valutazione in questo caso è binaria: vale 1 se viene riscontrata una corrispondenza, 0 altrimenti.

Gli agenti chiamati "Bayesian Filter Agent" si occupano invece dell'analisi dinamica dei messaggi. L'approccio dinamico si basa su algoritmi bayesiani e prende in considerazione il contenuto dei messaggi. Come accade per tutti i filtri bayesiani, un Bayesian Filter Agent necessita di un periodo iniziale di addestramento durante il quale viene generato il vocabolario di termini che sarà poi via via aggiornato ed incrementato. Proprio per il fatto che tale vocabolario deve necessariamente essere adattato sul singolo utente, ogni utente del sistema ha un proprio Bayesian Filter Agent specifico. Utilizzando algoritmi probabilistici Naive Bayes, questi agenti calcolano per ogni messaggio un valore legato alla probabilità che tale messaggio sia da considerarsi come spam e, in modo analogo all'analisi statica, inviano il valore calcolato agli agenti Analysis Manager. Questo valore di probabilità, chiamato "dynamic score", spazia fra 0 e 1. Per l'implementazione dell'analisi dinamica e dei Bayesian Filter Agent sono state utilizzate le librerie Classifier4J library [7].

Dopo le due analisi (statica e dinamica), sulla base dello static score e del dynamic score ricevuti, gli Analysis Managar calcolano una valutazione globale (detta "final score") e, in base al valore ottenuto, classificano il messaggio come ham o come spam-presumed e comunicano questa decisione all'agente EPA responsabile di quel messaggio. La valutazione globale del messaggio viene calcolata utilizzando la seguente formula (in cui *FS* sta per final score, *SS* per static score e *DS* per dynamic score).

$$FS = a \cdot SS + b \cdot DS$$

I valori dei parametri *a* e *b* della precedente formula vengono decisi dall'amministratore del sistema. Durante una prima fase di testing si è deciso di utilizzare $a = 0.7$ e $b = 0.3$, dando così più peso all'analisi dinamica. Il messaggio viene classificato come spam-presumed se il valore finale *FS* è superiore ad una certa soglia (un valore plausibile potrebbe essere *0.6*), impostabile sempre dall'amministratore del sistema.

Gli agenti chiamati "User Profile Manager", come nel caso del sistema RAP, sono i responsabili della gestione dei profili utente. In questo caso abbiamo due tipologie di profili:

- profili degli utenti del sistema: ogni utente di CAFE è legato ad un valore di affidabilità nella segnalazione dello spam ricevuto. Tale valore viene incrementato ogni volta che l'utente segnala un messaggio di spam ricevuto, mentre viene pesantemente diminuito se il messaggio segnalato come spam è invece considerato legittimo dalla maggior parte degli utenti del sistema. Nel momento il cui il valore di affidabilità diventa negativo, le segnalazioni di spam da parte dell'utente vengono sostanzialmente ignorate dal sistema, per questo motivo è fondamentale che gli utenti del sistema si comportino in modo corretto in relazione alla segnalazione dei messaggi illegittimi;
- profili dei contatti, relativamente ad ogni utente: è fondamentale che il sistema sia a conoscenza dei contatti di ciascun utente per operare una classificazione dei messaggi considerati legittimi (ham) nelle tre tipologie a cui si è precedentemente accennato (messaggi da utenti conosciuti, messaggi da utenti totalmente sconosciuti, messaggi da utenti conosciuti da almeno un utente del sistema). Inoltre, ogni utente può assegnare un valore di reputazione ai propri contatti in modo che il sistema possa classificare, sulla base dei mittenti, i messaggi anche in termini di rilevanza.

L'agente chiamato "Starter Agent" si occupa di attivare gli agenti Personal Agent e Byesian Filter Agent nel momento in cui l'utente collegato ad essi diventa attivo nel sistema e richiede, tramite il proprio client e-mail, la ricezione dei nuovi messaggi di posta elettronica.

L'agente chiamato "Directory Facilitator" è un agente JADE che ha il compito di comunicare ad un agente che lo richiede l'indirizzo di un agente attivo nel sistema che fornisce un determinato servizio.

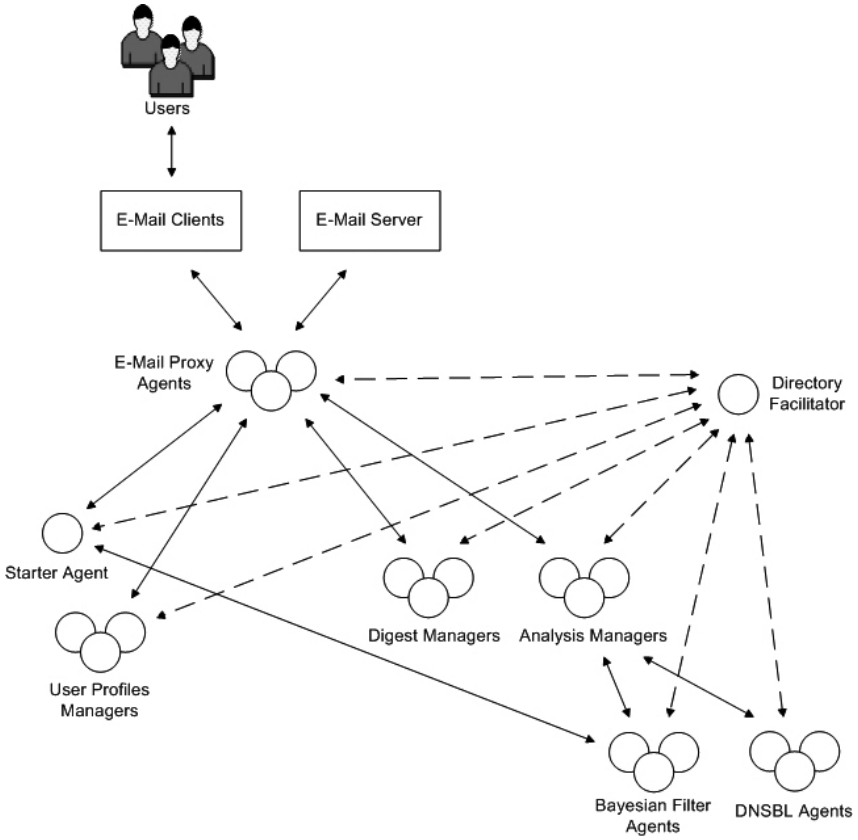

Figura 1.7 - Architettura del sistema CAFE

La figura 1.7 mostra una schematizzazione dell'architettura del sistema CAFE, sottolineando le varie interazioni fra le differenti tipologie di agenti. Come nel caso

di RAP, anche una piattaforma CAFE può essere distribuita su diversi nodi computazionali. Inoltre, in figura 1.7 le immagini che mostrano la sovrapposizione di più di un utente/agente suggeriscono che nel sistema può essere presente contemporaneamente più di un utente/agente di quel tipo.

1.5.5 Funzionamento del Sistema

Prima di descrivere in modo completo il funzionamento di CAFE analizzando nel dettaglio il processo completo di analisi e catalogazione dei messaggi di posta elettronica, è necessario fare una breve considerazione: come già affermato nella sezione 1.5.3, il sistema CAFE classifica i messaggi e-mail in tre categorie distinte: ham (messaggi legittimi), spam e spam-presumed. Questo perché in fase di progettazione del sistema si è supposto che una classificazione "binaria" dei messaggi in spam e ham fosse troppo restrittiva, soprattutto nel caso in cui un messaggio superasse la prima fase di analisi (basata sul digest) e dovesse essere sottoposto a tecniche di analisi più classiche (basate sugli approcci statico e dinamico descritti nella precedente sezione), che spesso non raggiungono un'elevata efficacia. Per questo motivo, una e-mail viene classificata come spam solo se viene riconosciuta tale dal primo livello di analisi, ossia se viene riscontrata una corrispondenza fra il suo digest ed uno dei digest di messaggi già considerati spam dal sistema. Se invece una e-mail viene segnalata come spam dal secondo livello di analisi, ossia dall'agente Analysis Manager, essa viene classificata dal sistema come spam-presumed. Dopo che i messaggi sono stati classificati da parte del sistema nelle tre categorie, gli utenti hanno il compito fondamentale di controllare i messaggi etichettati come spam-presumed e, per ogni messaggio, informare il sistema sull'effettiva natura del messaggio (ham o spam). In modo analogo, gli utenti hanno anche il dovere di notificare al sistema la presenza di messaggi legittimi tra quelli catalogati come spam e viceversa.

La descrizione del funzionamento del sistema può essere suddivisa nei seguenti step:

1) log-in dell'utente ed attivazione del relativo E-mail Proxy Agent;
2) analisi basata sul digest del messaggio;

3) analisi statica/dinamica (se necessaria);
4) classificazione del messaggio;
5) valutazione dell'utente.

Log-in dell'utente ed attivazione del relativo E-mail Proxy Agent: quando un utente, tramite il proprio client e-mail, richiede la ricezione dei nuovi messaggi, lo Starter Agent attiva immediatamente gli agenti E-mail Proxy Agent e Bayesian Filter Agent associati all'utente stesso. L'agente EPA, sulla base delle credenziali di accesso ricevute dal client e-mail dell'utente, si collega al server di posta elettronica, scarica il messaggio (supponiamo per semplicità che la casella e-mail dell'utente contenga un solo nuovo messaggio) e calcola il digest del messaggio.

Analisi basata sul digest del messaggio: il digest del messaggio viene inviato ad un agente Digest Manager, che si occupa della prima fase di analisi. Se viene riscontrata corrispondenza fra il digest del messaggio ed uno dei digest riconosciuti dal sistema come spam, il messaggio viene immediatamente etichettato come spam dal Digest Manager (ad es. inserendo la scritta "SPAM" all'interno dell'oggetto del messaggio) e viene inviato al client e-mail dell'utente: in questo caso la fase di analisi ha termine. Se invece non viene riscontrata alcuna corrispondenza, è necessaria la seconda fase di analisi.

Analisi statica/dinamica (se necessaria): il messaggio viene inviato all'agente Analysis Manager, responsabile della seconda fase di analisi. A questo punto, il messaggio viene analizzato utilizzando due differenti approcci: un approccio statico basato su database DNSBL (DNSBL Agent) ed un approccio dinamico basato su filtro bayesiano (Bayesian Filter Agent). Sulla base dei risultati ottenuti, gli agenti DNSBL Agent e Bayesian Filter Agent comunicano all'agente Analysis Manager le due valutazioni del messaggio (static e dynamic score). Combinando le due valutazioni, utilizzando un approccio "a soglia", l'agente Analysis Manager cataloga il messaggio come ham o come spam-presumed e comunica tale risultato finale all'agente E-mail Proxy Agent responsabile del messaggio. Come nel caso dello spam, se il messaggio è stato catalogato come spam-presumed, viene etichettato come tale (ad es. inserendo la scritta "SPAM-PRESUMED" all'interno

dell'oggetto del messaggio), mentre se il messaggio è stato catalogato come ham, esso viene inviato inalterato al client e-mail dell'utente.

Classificazione del messaggio: se il messaggio è stato catalogato dal sistema come ham, ossia come messaggio legittimo, l'agente E-mail Proxy Agent procede alla sua classificazione a seconda del mittente del messaggio stesso in una delle tre seguenti categorie: messaggi da utenti conosciuti, messaggi da utenti sconosciuti o messaggi da utenti conosciuti da altri utenti del sistema. Nell'ultimo caso, al messaggio viene associato un livello di rilevanza sulla base della reputazione associata all'utente-mittente da parte degli utenti del sistema che lo hanno tra i propri contatti. Questo processo di classificazione richiede il supporto degli agenti User Profile Manager: l'agente E-mail Proxy Agent comunica l'indirizzo e-mail del mittente del messaggio da un agente UPM e questo gli comunica se l'utente è conosciuto o meno dall'utente destinatario o se l'utente è presente fra i contatti di almeno uno degli utenti del sistema (con la relativa reputazione, se disponibile). Grazie a questa classificazione, un utente può assegnare un maggiore livello di importanza ai messaggi ricevuti da mittenti conosciuti e, successivamente, associare ai restanti messaggi un'importanza dipendente dalla reputazione associata del mittente.

Come detto in precedenza, il risultato del processo globale di analisi di CAFE consiste nella modifica o meno dell'oggetto del messaggio e-mail. In questo modo, l'utente, creando una semplice regola di gestione della posta recapitata sulla base dell'oggetto dei messaggi, può far sì che il proprio client e-mail cataloghi in modo automatico la posta in arrivo.

Valutazione dell'utente: essendo CAFE un sistema che si basa totalmente sulla vena collaborativa della propria comunità, è chiaro che diventa molto importante il comportamento di ogni singolo utente, al quale è richiesto l'invio al sistema di un feedback sui risultati ottenuti. Come vedremo nella sezione successiva, se il livello di collaborazione degli utenti è alto, l'efficacia del sistema diventa massima molto rapidamente. I compiti degli utenti del sistema sono fondamentalmente due: l'assegnazione di un livello di reputazione ai propri contatti e la classificazione (e la relativa notifica al sistema) dei messaggi ricevuti come spam-presumed. Gli

utenti sono inoltre in dovere di notificare al sistema eventuali errori di catalogazione riscontrati all'interno delle categorie ham e spam. Scendendo maggiormente nel dettaglio, l'utente, durante il proprio processo di valutazione, può imbattersi nelle seguenti situazioni:

4.1) un messaggio etichettato come spam-presumed è spam;

4.2) un messaggio etichettato come spam-presumed è ham;

4.3) un messaggio etichettato come ham è spam;

4.4) un messaggio etichettato come spam è ham.

Un messaggio etichettato come spam-presumed è spam: il digest del messaggio viene inserito nel database dei digest riconosciuti come spam, così che messaggi simili a questo in futuro siano catalogati come spam. Inoltre il comportamento del Bayesian Filter Agent viene di conseguenza modificato.

Un messaggio etichettato come spam-presumed è ham: il comportamento del Bayesian Filter Agent viene di conseguenza modificato in modo che in futuro un messaggio simile a questo sia trattato come ham.

Un messaggio etichettato come ham è spam: come nel primo caso, il digest del messaggio viene inserito nel database dei digest riconosciuti come spam ed il comportamento del Bayesian Filter Agent viene di conseguenza modificato.

Un messaggio etichettato come spam è ham: se il messaggio è stato etichettato come spam, significa che il suo digest è presente nel database dei digest riconosciuti come spam. Di conseguenza il digest del messaggio viene immediatamente eliminato dal database. Anche in questo caso, il comportamento del Bayesian Filter Agent viene modificato.

Dal momento che l'efficacia del sistema è fortemente condizionata dalle valutazioni fornite dagli utenti, gli agenti User Profile Manager associano ad ogni utente un livello di affidabilità. Tale livello viene modificato sulla base del comportamento degli utenti stessi e, se tale livello diventa negativo, il sistema tiene comunque conto di eventuali feedback dell'utente (è possibile che il livello di affidabilità in futuro ritorni maggiore di zero), ma non agisce di conseguenza. Ad esempio, se un utente con livello di affidabilità negativo informa il sistema che uno dei messaggi ricevuti come ham è in realtà spam, il digest del messaggio non viene

inserito nel database dei digest riconosciuti come spam fino a quando la reputazione dell'utente non ritorna ad essere positiva.

1.5.6 Analisi delle Prestazioni

In questa ultima sezione presentiamo un'analisi matematica del processo di filtraggio dello spam in CAFE. L'efficacia del sistema nell'individuazione di un messaggio illegittimo viene analizzata in funzione del numero di utenti che ricevono quel messaggio e del livello di collaborazione degli utenti stessi. Tramite la trattazione matematica si giungerà alla definizione di un upper bound e di un lower bound per il valore della probabilità di ottenere un falso negativo (ossia della probabilità che CAFE classifichi il messaggio illegittimo come ham anziché come spam).

Prima di iniziare l'analisi, è necessario introdurre alcune variabili per avere un modello matematico del sistema: con n indichiamo il numero complessivo di utenti di CAFE, mentre con x la probabilità che un nuovo messaggio illegittimo non sia riconosciuto come tale dall'analisi di secondo livello (DNSBL Agent e Bayesian Filter Agent). In altre parole, x identifica la probabilità di fallimento da parte dei metodi classici di individuazione dello spam (analisi statica ed analisi dinamica). Infine, con la variabile p indichiamo la probabilità di ottenere un falso negativo, ossia la probabilità che CAFE cataloghi un messaggio illegittimo come ham.

Almeno inizialmente, per semplificare l'analisi si suppone che ogni utente del sistema collabori in modo efficiente (ossia che ogni utente del sistema valuti ogni messaggio ricevuto ed informi il sistema di conseguenza) e che ogni messaggio di spam sia inviato a tutti gli utenti. Questa rappresenta la situazione migliore, in cui:

$$\frac{x}{n} \leq p \leq x \qquad (1)$$

In accordo con la precedente formula, il valore di p è massimo se il sistema ha un unico utente ($n = 1$). Questo è ovvio, dal momento che se la comunità consiste in un unico utente, l'aspetto collaborativo del sistema scompare e CAFE si

trasforma in un classico sistema di classificazione delle e-mail basato sugli approcci statico e dinamico. Quindi in questo caso è corretto che la probabilità *p* di ottenere un falso negativo coincida con *x*. Se il numero degli utenti aumenta, il valore di *p* diminuisce in modo proporzionale alla crescita del numero *n* di utenti del sistema.

Ovviamente questa rappresenta la situazione ideale, in quanto è plausibile supporre che un messaggio di spam non sia inviato all'intera comunità di CAFE, ma solamente ad un sottoinsieme di utenti. Per questo motivo introduciamo una nuova variabile, *r*, che rappresenta il numero di utenti che effettivamente ricevono il nuovo ipotetico messaggio di spam. Il valore di *r* è normalizzato rispetto a *n*:

$$\frac{1}{n} \leq r \leq 1 \qquad (2)$$

Dalla precedente formula, possiamo dedurre che il valore di *r* è *1*/*n* se il messaggio di spam è indirizzato solo ad un utente, mentre è *1* se il messaggio è ricevuto da tutti gli utenti del sistema. Con l'introduzione di questo nuovo parametro, il lower bound della variabile *p* cambia e la formula (1) diventa:

$$\frac{1}{r} \cdot \frac{x}{n} \leq p \leq x \qquad (3)$$

In accordo con la formula (3), se il nuovo messaggio di spam viene ricevuto dall'intera community (*r* = *1*), si ritorna al caso ideale descritto dalla formula (1). Se invece il messaggio di spam raggiunge solo uno degli utenti (*r* = *1*/*n*), si ottiene *p* = *x* e l'aspetto collaborativo di CAFE viene a cadere.

Fino a questo punto si è supposto che ogni utente di CAFE sia disposto a prestare la propria collaborazione, informando il sistema ogni qual volta riceva un messaggio illegittimo. A questo punto è necessario considerare la possibilità che uno o più utenti non siano disposti a collaborare, causando così una diminuzione

dell'efficacia del sistema. Indicando con *n'* il numero di utenti non disposti a collaborare, è possibile scrivere il seguente sistema:

$$\begin{cases} (1+n')\cdot\frac{1}{r}\cdot\frac{x}{n}\le x \\ n'\ge 0 \end{cases} \qquad (4)$$

Semplificando la prima equazione si ottiene:

$$\begin{cases} n'\le r\cdot n-1 \\ n'\ge 0 \end{cases} \Rightarrow 0\le n'\le r\cdot n-1 \qquad (5)$$

In accordo con la formula (2), il valore massimo del parametro *r* è *1*. In questo caso (*r = 1*) possiamo scrivere:

$$0\le n'\le n-1 \qquad (6)$$

A questo punto possiamo considerare nuovamente il sistema (5): se il nuovo messaggio di spam è indirizzato ad un solo utente, è valida la seguente formula:

$$\begin{cases} r=\frac{1}{n} \\ 0\le n'\le r\cdot n-1 \end{cases} \Rightarrow n'=0 \qquad (7)$$

Tuttavia, se il messaggio illegittimo è indirizzato all'intera comunità di utenti di CAFE, la formula ottenuta è la (6).

Con l'introduzione del parametro *n'*, è possibile ottenere un modello matematico completo del processo di individuazione dello spam, come mostrato dalla seguente formula.

$$(1+n')\cdot\frac{1}{r}\cdot\frac{x}{n}\leq p\leq x \qquad (8)$$

Con la formula (8) siamo giunti alla definizione di un upper bound e di un lower bound per il valore della probabilità che CAFE cataloghi un nuovo messaggio illegittimo come ham. Nella situazione ideale, in cui ogni utente del sistema sia disposto a collaborare (*n' = 0*) ed in cui i nuovi messaggi di spam siano inviati a tutta la comunità (*r = 1*), la probabilità *p* di avere un falso negativo raggiunge il suo valore minimo, pari a *x*/*n*. Sulla base di quest'ultimo risultato, è possibile affermare che l'efficacia di CAFE nell'individuazione dello spam cresca proporzionalmente al numero degli utenti del sistema e, ovviamente, al livello di collaborazione che ogni utente è disposto a fornire.

Il comportamento del sistema è stato valutato utilizzando un piccolo gruppo di utenti (*n = 10*) ed un test-set di messaggi comprendente sia e-mail legittime che e-mail di spam. In primo luogo è stata valutato il sistema nella situazione non-collaborativa (*r = 1/n*, *n' = 0*): in questo caso l'efficacia di CAFE è paragonabile a quella di un classico filtro antispam, raggiungendo una probabilità di falsi negativi vicina al 5%. Successivamente è stato valutato il sistema nella sua natura collaborativa, aumentando i valori di *r* ed *n'* e dimostrando così nella pratica la trattazione matematica presentata il precedenza: nella condizione ideale la probabilità di avere falsi negativi, nel caso di test, è stata inferiore allo 0.5%.

2
Voice Over IP

Questo capitolo presenta l'attività di ricerca incentrata sulla tecnologia Voice Over IP. Inizialmente vengono descritti nel dettaglio i due maggiori standard VoIP, H.323 e SIP, e, dopo un'analisi comparata dei due, viene presentata l'attività di ricerca e sviluppo basata sul protocollo SIP.

2.1 Premessa sul Voice Over Packet

Con Voice over Packet si intende l'insieme di studi portati avanti negli ultimi anni sulla trasmissione di audio e video su reti a commutazione di pacchetto. I due esempi più significativi sono il Voice Over IP (rivolto alle reti private) e la telefonia IP (rivolta invece alle reti IP pubbliche).

Il desiderio di utilizzare le reti a pacchetto per servizi real-time nasce di pari passo con lo sviluppo delle potenzialità tecniche dovuto al grosso miglioramento delle prestazioni dei processori, delle periferiche di rete dei PC ed ancora alla progressiva affermazione delle reti, sia pubbliche sia private, basate sul protocollo IP (Internet Protocol). Le aziende che avevano già a disposizione una propria rete privata hanno subito colto la novità riscontrando in essa la possibilità di diminuire i costi delle comunicazioni interne e di integrare servizi supplementari come la videoconferenza. Da un punto di vista più ampio, lo sviluppo delle applicazioni real-time su reti a pacchetto può essere ormai inquadrato in una richiesta globale del mercato o delle aziende di una rete integrata nei servizi.

Nonostante la rapida affermazione, esistono diversi problemi legati a questa scelta dovuti per la maggior parte al fatto che il protocollo IP, così come la rete Internet, sono nati per la distribuzione esclusiva di dati. Infatti la gestione dei pacchetti da parte del protocollo IP comporta una ricomposizione di questi nell'ordine originale. Se alcuni pacchetti subiscono errori di trasmissione o ritardi vari si capisce come il processo di ricomposizione possa risultare rallentato. Per applicazioni classiche, come ad esempio la posta elettronica, tale ritardo non acquista particolare significato, ma per applicazioni come la telefonia, dove l'interazione real-time tra le parti è fondamentale, questo può provocare un degradamento notevole della qualità del servizio (QoS). Nella telefonia IP, per cercare di far fronte almeno in parte a questo problema, viene utilizzato il protocollo di trasporto UDP (User Datagram Protocol), semplice ma meno affidabile, per il trasporto del traffico vocale, mentre il TCP (Transmission Control Protocol), più complesso e sicuro, è utilizzato per garantire l'affidabilità del messaggio di instaurazione di una chiamata.

Oltre alla QoS, esistono altri problemi legati al Voice over Packet, tra i più importanti possiamo citare:

- scalabilità e impatto sulle reti IP;
- integrazione con la rete PSTN e con le reti mobili;
- sicurezza;
- compatibilità.

Almeno ad una parte di questi cercò di dare una risposta la ITU (International Telecommunications Union) emanando una raccomandazione finalizzata ad armonizzare i vari protocolli utilizzati e a fornire una base per lo sviluppo di applicazioni real-time su reti a pacchetto. Il risultato fu nel 1996 la prima versione dello standard H.323.

Benché H.323 abbia avuto un discreto successo soprattutto nei primi anni, il protocollo mostra diverse limitazioni legate in modo particolare all'eccessiva complessità e soprattutto alla scarsa flessibilità. Di conseguenza, a partire dal 1999, ha preso il via, per iniziative della IETF (Internet Engineering Task Force), un progetto finalizzato allo sviluppo di una nuova modalità semplice e modulare per

gestire le comunicazioni VoIP. Il risultato è stato il protocollo SIP (Session Initiation Protocol), il quale, a differenza di H.323 che comprende una suite di protocolli completa, non rappresenta un sistema per le comunicazioni integrato verticalmente, ma piuttosto può essere usato come componente in altri protocolli IETF per costruire un'architettura multimediale completa: SIP non fornisce servizi, ma primitive per implementarli.

2.2 Lo Standard H.323

La famiglia di standard (o raccomandazioni) ITU H.32x è finalizzata alle comunicazioni di tipo multimediale su differenti tipi di reti:

- H.324 per la rete PSTN;
- H.320 per la rete N-ISDN;
- H.321 e H.310 per la rete B-ISDN;
- H.322 per reti LAN che forniscono QoS garantita.

A questi si affianca, nel 1996, lo standard H.323, inizialmente ideato per la gestione di comunicazioni multimediali su reti LAN con QoS non garantita, come dimostra il titolo della prima versione: "H.323 - Visual telephone system and equipment for LANs that provide a non guaranteed quality of service". Due anni più tardi, con la creazione di una versione più completa dal titolo "H.323 - Packet based multimedia communication system", lo standard H.323 assume una connotazione più generale, fornendo una descrizione dei terminali, dei dispositivi e dei servizi per comunicazioni multimediali (audio, video, dati) su reti a pacchetto.

La raccomandazione H.323 viene detta "ad ombrello" in quanto, più che introdurre nuovi protocolli o nuove codifiche, essa tenta di armonizzare le indicazioni dei precedenti standard della serie H e di prevedere tutta una famiglia di gateway idonei a garantire l'interoperabilità di queste con le entità di videotelefonia in Internet. Per ottenere tale interoperabilità, la pila protocollare definita con H.323 deve operare al di sopra dello strato di trasporto della rete sottostante, così che i protocolli H.323 possano essere utilizzati con una qualsiasi rete a pacchetto.

In questo senso, la raccomandazione H.323 stabilisce quattro tipi di componenti standard, detti "H.323 Entity", che definiscono in tutto e per tutto una sottorete, a livello applicativo, interagente con le altre tipologie di rete:

- Terminali;
- Gateway;
- Gatekeeper;
- Multipoint Control Unit (MCU).

2.2.1 Componenti H.323

I **Terminali** costituiscono gli endpoint in cui si originano e terminano i flussi di dati e le segnalazioni H.323. Possono essere PC, PDA, telefoni IP o un qualsiasi dispositivo dedicato che contenga l'implementazione dei protocolli H.323. Tutti i terminali garantiscono obbligatoriamente comunicazioni audio e facoltativamente video e dati.

Il **Gateway** è un elemento non obbligatoriamente presente in una conferenza H.323, ma necessario ogni volta che occorre mettere in comunicazione reti tra loro eterogenee. La funzione del gateway consiste proprio nella traduzione dei protocolli delle due reti, ad esempio H.225 - H.245 verso H.221 - H.242 nel caso di un gateway che operi tra una LAN e una rete PSTN. Inoltre il gateway opera una traduzione dei codec audio/video e si occupa del setup e dell'abbattimento della chiamata su entrambe le reti che mette in contatto.

Il **Gatekeeper**, analogamente al gateway, non rappresenta un elemento obbligatoriamente presente in una comunicazione multimediale, tuttavia, se utilizzato, svolge numerose importanti funzioni e offre tutta una serie di servizi a cui altrimenti si deve rinunciare. Il gatekeeper agisce come punto di riferimento per tutte le chiamate all'interno della sua zona, fornendo servizi di controllo di chiamata agli endpoint registrati. In particolare, le funzionalità che il gatekeeper deve garantire sono le seguenti:

- address translation: traduzione degli indirizzi alias usati sulla LAN in indirizzi IP. Per far ciò, il gatekeeper è in possesso di una tabella delle

registrazioni continuamente aggiornata attraverso i messaggi del protocollo RAS (Registration Admission Status);
- admission control: controllo ed ammissione di un endpoint in una zona H.323. L'ammissione può essere basata su autorizzazioni alla chiamata, sulla banda richiesta o su altri criteri;
- bandwidth control: controllo della banda;
- zone management: controllo dei componenti H.323 appartenenti alla zona H.323 di competenza del gatekeeper.

La **Multipoint Control Unit** (MCU) è un componente H.323 che permette a tre o più terminali di partecipare a una conferenza multipunto. Infatti la MCU riceve i flussi audio e video dai partecipanti alla conferenza e restituisce a tutti un unico flusso contenete i contributi di ognuno.

2.2.2 Protocolli H.323

Come già anticipato, H.323 rappresenta uno standard "ad ombrello" che racchiude e collega insieme una serie di protocolli applicativi, specificati poi nel dettaglio in altre raccomandazioni. I più importanti protocolli appartenenti allo standard H.323 sono i seguenti:

- G.7xx - Insieme di codifiche audio;
- H.26x - Insieme di codifiche video;
- T.120 - Protocollo per la trasmissione dati;
- RTP/RTCP - Protocolli di trasmissione di flussi real-time;
- H.225 RAS - Registration Admission Status;
- H.225/Q.931 Call Signalling - Instaurazione della chiamata;
- H.245 Control Signalling - Contrattazione della tipologia della chiamata.

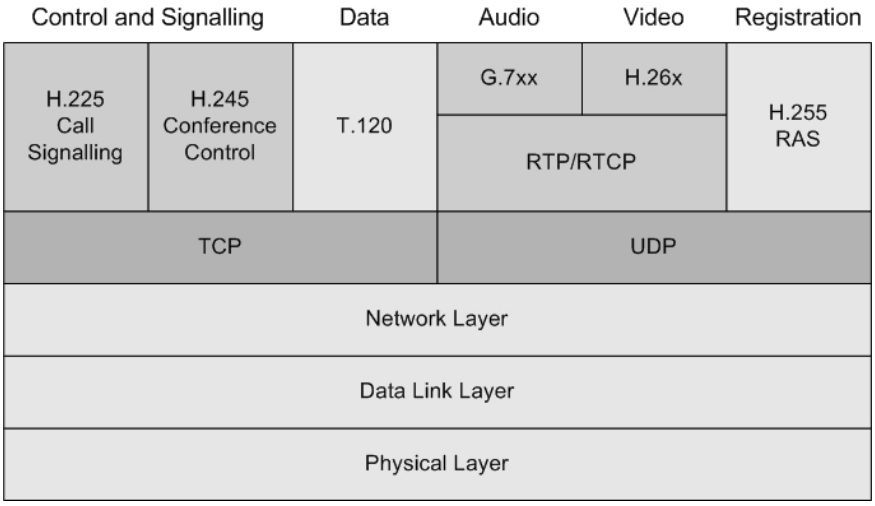

Figura 2.1 - Stack protocollare di H.323

Di seguito e nella sezione successiva verranno presentati nel dettaglio le codifiche ed i protocolli compresi nello standard H.323.

La **codifica audio**, supportata obbligatoriamente da qualsiasi terminale H.323, comprende vari protocolli, definiti dalla ITU nella serie G.7xx. Vediamone alcuni:

- G.711: codifica PCM a 64 Kbps usata nella PSTN (obbligatoria per qualunque terminale);
- G.722: codifica ADPCM a 7 KHz con 48/56/64 Kbps;
- G.723: tecnica di compressione che arriva ad una bit-rate di 5.3/6.3 Kbps basandosi su MP-MLQ (Multipulse, Multilevel Quantization);
- G.728 e G.729: codifiche a basso ritardo CELP (Code Excited Lincar Prediction) con bit-rate di 8/16 Kbps;
- MPEG-1;
- GSM: codifica usata nello standard di telefonia mobile a 13 Kbps.

I supporti per le **codifiche video** sono stati definiti dalla ITU con le sigle H.261 ed H.263. Vediamoli in dettaglio:

- H.261: standard di comunicazione di immagini in movimento molto simile all'MPEG, tramite il quale è possibile una trasmissione dati con

un'occupazione di banda che può scendere fino alla bit-rate minima di 64 Kbps, ovvero un canale base ISDN;

- H.263: supporta trasmissioni a minore bit-rate senza perdita di qualità, ma comporta ovviamente un maggior carico elaborativo integrando la codifica H.261 con tecniche di predizione dei frame.

Lo **standard T.120** si occupa di specificare i protocolli per la trasmissione in tempo reale di dati. Come l'H.323, anche il T.120 è una raccomandazione "ad ombrello" che racchiude un insieme di standard al fine di ottenere uno scambio di dati tra terminali appartenenti a reti diverse. Nell'ambito H.323, il trasferimento dati mantiene una certa indipendenza dai flussi audio/video di una conferenza, in quanto una trasmissione T.120 può instaurarsi sia indipendentemente, tramite una preventiva instaurazione della chiamata, sia durante una preesistente chiamata audio/video. Inoltre la terminazione di una connessione T.120 non implica la fine di una chiamata.

Nel momento in cui si è voluto usare reti a pacchetto per le applicazioni in tempo reale, ci si è subito resi conto che il protocollo di strato di trasporto più adatto da usare era UDP. Tuttavia, se da un lato questo permetteva una maggiore trasparenza temporale, dall'altro rendeva la trasmissione totalmente inaffidabile. I **protocolli RTP e RTCP** sono protocolli di livello applicativo sviluppati dalla Internet Engineering Task Force (IETF), i quali mirano ad aggiungere funzionalità ai protocolli di strato di trasporto. Spesso vengono introdotti come protocolli di trasporto perché realizzano funzioni end-to-end per le varie applicazioni multimediali sovrastanti in modo indipendente da quest'ultime.

Infine **RAS** è l'acronimo di "Registration Admission Status" ed è il protocollo che definisce i messaggi per lo scambio di segnalazione di controllo tra endpoint (terminali, MCU, gateway) e gatekeeper per effettuare la registrazione. I messaggi RAS vengono scambiati con una canale RAS che utilizza il protocollo di trasporto UDP. Questo canale è aperto indipendentemente da una chiamata ed è quindi aperto prima di qualsiasi altro canale (nel caso di utilizzo del gatekeeper).

2.2.3 Struttura della Chiamata

Il **protocollo H.255** definisce i messaggi scambiati al fine di instaurare una chiamata fra endpoint. Questi messaggi sono scambiati attraverso il canale H.225 di segnalazione (Call Signalling Channel).

Se nella rete H.323 non è presente un gatekeeper, i messaggi sono scambiati direttamente tra i due endpoint interessati dalla chiamata.

Il ciclo di vita di una comunicazione si articola nelle seguenti fasi:

A) call setup;
B) initial communication and capability exchange;
C) establishment of audiovisual communication;
D) call services;
E) call termination.

Il protocollo H.225 si occupa della prima fase di **call setup**, che riguarda l'instaurazione della chiamata. Grazie alla flessibilità dei componenti e dell'architettura di una rete H.323, sono possibili diversi scenari per l'instaurazione di una chiamata.

Vediamo nel dettaglio la situazione più semplice (v. figura 2.2), che vede i due endpoint coinvolti nella comunicazione non registrati presso alcun gatekeeper. In questo caso i messaggi di controllo e di segnalazione vengono scambiati direttamente tra gli endpoint, evitando messaggi RAS. Questo impone che l'endpoint chiamante conosca a priori l'indirizzo dell'endpoint chiamato.

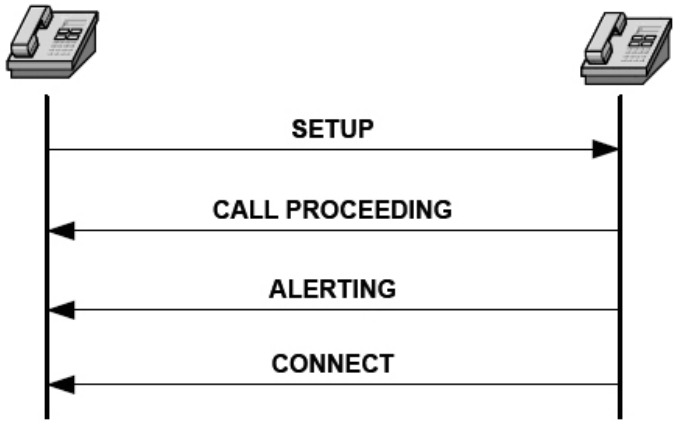

Figura 2.2 - Call setup in assenza di gatekeeper

L'endpoint 1 (chiamante) invia il messaggio SETUP verso un ben noto Call Signalling Channel TSAP Identifier dell'endpoint 2. L'endpoint 2 risponde con CONNECT, che contiene il Control Channel Transport Address da usare per la segnalazione H.245. Il messaggio di CALL PROCEEDING avverte che l'endpoint chiamato sta iniziando le procedure di connessione, mentre il messaggio di ALERTING ha una funzione di wait nel caso in cui CONNECT venga spedito dopo 4 secondi da Setup.

Una volta che entrambi gli endpoint hanno scambiato i messaggi di setup completando la fase A, si passa alla fase B (**initial communication and capability exchange**) in cui i due endpoint aprono il **canale di comunicazione H.245** utilizzato per lo scambio dei segnali di controllo della chiamata.

Il controllo di funzionamento H.245 usa il canale di controllo H.245 (Control Signalling Channel) per trasportare end-to-end i messaggi di controllo che governano le operazioni delle entità, inclusi scambio di capacità, apertura e chiusura dei canali logici, richieste di modi preferenziali di funzionamento, messaggi di controllo di flusso e altri comandi. La segnalazione H.245 è stabilita fra due endpoint, fra un endpoint ed una MCU o fra un endpoint ed un gatekeeper. Un endpoint deve aprire un canale H.245 per ogni chiamata a cui sta partecipando,

di conseguenza una MCU o un gatekeeper, che possono supportare molte chiamate contemporaneamente, devono aprire più canali H.245 di controllo.

Il primo messaggio H.245 che deve essere scambiato è quello sulle capacità degli endpoint. Vengono definite in questo contesto le capacità sia trasmissive che ricettive, fornendo all'endpoint col quale si vuole iniziare la comunicazione le possibili alternative in termini di modalità di codifica e bit-rate degli stream audio/video o dati. Successivamente l'endpoint supporta la determinazione master-slave utile per definire quale MCU sarà attiva nella conferenza. Questo nel caso in cui più endpoint vogliono dare vita ad una conferenza.

Se una delle due procedure fallisce, l'endpoint abbandona la connessione e si entra nella fase E (call termination). Altrimenti, in caso di successo, si passa alla fase C (**establishment of audiovisual communication**). In questa fase le procedure H.245 vengono usate per aprire i canali logici per lo scambio dei vari flussi informativi.

La fase D (**call services**) della comunicazione racchiude alcuni servizi specifici che possono essere richiesti durante la chiamata, come la modifica della banda utilizzabile o la richiesta di informazioni su uno specifico endpoint da parte del gatekeeper.

L'ultima fase (**call termination**) è quella di terminazione della chiamata (che può essere richiesta in qualsiasi momento da uno degli endpoint coinvolti nella comunicazione) e della conseguente chiusura dei canali logici di scambio dati.

2.3 SIP - Session Initiation Protocol

SIP, acronimo di "Session Initiation Protocol", è un protocollo di segnalazione il cui standard è definito dalla IETF per stabilire connessioni real-time e conferenze su reti IP. Ogni sessione può includere diversi tipi di flussi dati come audio e video, sebbene ora la maggior parte delle estensioni SIP riguardino le comunicazioni audio. È un protocollo di tipo testuale e si avvicina molto ad HTTP (HyperText Transfer Protocol) come struttura, anche se a livello di trasporto utilizza UDP anziché TCP. SIP è indipendente dal tipo di rete su cui si appoggia, è stato

strutturato per essere uno standard aperto e scalabile, in modo che possa essere utilizzato per diversi scopi, tra cui principalmente la telefonia IP, la videoconferenza, la chat, ma anche lo streaming di eventi live.

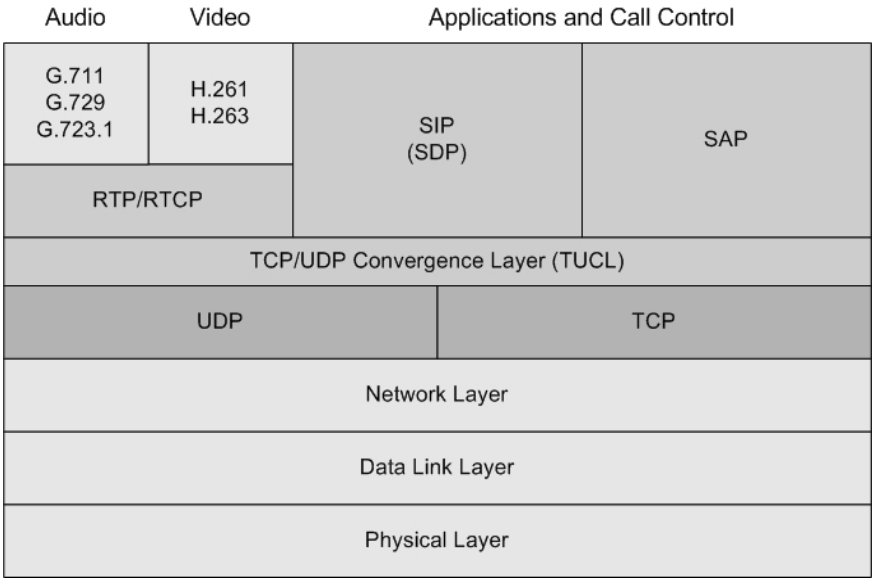

Figura 2.3 - Stack protocollare di SIP

Come è possibile notare dalla figura 2.3, anche in SIP, come in H.323, il trasporto dei media è affidato ad RTP, ed anche le codifiche audio/video supportate sono sostanzialmente le stesse. Il protocollo SDP (Session Description Protocol), sviluppato come SIP dalla IETF, definisce lo standard per la descrizione delle sessioni SIP. Si vedranno alcuni esempi della sintassi SDP più avanti quando verranno analizzati i messaggi SIP.

A differenza dello standard H.323, SIP è stato creato appositamente per Internet. Oltre all'instaurazione, alla modifica ed alla terminazione delle chiamate, tramite SIP è anche possibile invitare nuovi partecipanti a sessioni già attive o far sì che una sessione attiva supporti nuovi canali di comunicazione. Uno degli aspetti fondamentali di SIP consiste nel fatto che il protocollo supporta in modo del tutto trasparente all'utente servizi di "name mapping" e di redirezione delle chiamate,

consentendo in questo modo la completa mobilità dell'utente stesso, che mantiene un'unica costante interfaccia SIP completamente slegata dalla propria posizione fisica sulla rete.

Vediamo alcuni degli aspetti chiave di SIP che consentono un'agevole gestione delle chiamate:

- user location: individuazione del terminale verso il quale inviare la chiamata;
- user availability: valutazione della disponibilità dell'utente nel ricevere l'eventuale chiamata;
- user capabilities: individuazione dei canali di comunicazione da utilizzare;
- session setup: individuazione dei parametri di sessione da utilizzare per entrambe le parti coinvolte;
- session management: gestione e terminazione della sessione ed invocazione dei servizi.

Dal punto di vista architetturale ed implementativo, essendo SIP un protocollo text-based che non presenta un elevato livello di complessità, la fase di sviluppo risulta molto agevole anche utilizzando linguaggi di programmazione di alto livello come Java.

2.3.1 Struttura ed Architettura

Come detto in precedenza, SIP è un protocollo di tipo testuale ispirato ai protocolli HTTP e SMTP (Simple Mail Transfer Protocol), dei quali riprende la sintassi, il sistema di funzionamento e l'architettura di tipo client-server. Lo scopo di SIP è la gestione delle chiamate, l'attivazione delle sessioni e la chiusura delle stesse, SIP non si occupa del trasporto dei dati (audio/video) ma solo della segnalazione, la comunicazione vera e propria avviene attraverso altri protocolli come RTP. Nella fase di setup della chiamata vengono definite le caratteristiche della connessione da stabilire, queste informazioni si trovano all'interno del payload dei pacchetti (ossia la parte di dati dei pacchetti, che si affianca all'header, destinato invece a contenere i campi utilizzati per l'instradamento della chiamata e le informazioni di servizio del protocollo).

Le entità principali che interagiscono attraverso SIP sono le seguenti:

- **SIP User Agent** (**UA**): chiamato anche terminale, rappresenta un'estremità di una comunicazione, un dispositivo hardware o software che è in grado di iniziare e ricevere una chiamata utilizzando SIP come protocollo di segnalazione. Esistono programmi che possono simulare un terminale SIP in un PC (chiamati softphone) e dispositivi hardware simili a telefoni tradizionali, che invece di appoggiarsi ad una rete telefonica tradizionale (PSTN o ISDN), utilizzano una connessione IP. Gli UA possono funzionare come server (**UAS** - **User Agent Server**) o come client (**UAC** - **User Agent Client**) a seconda che ricevano o inizino una chiamata.
- **SIP Proxy Server**: si occupa di gestire le chiamate per una determinata area e di smistare e rielaborare le chiamate che gli arrivano secondo politiche definite. Normalmente gestisce tutti gli utenti/UA di un dominio SIP e smista le chiamate in arrivo verso i corretti UA per instaurare le comunicazioni.
- **SIP Redirect Server** (o **Registrar**): reindirizza le chiamate SIP verso altre destinazioni. La differenza sostanziale rispetto al Proxy Server è che questo non si occupa di inoltrare la chiamata all'indirizzo specificato, ma risponde alle richieste degli UA aggiornandoli sugli indirizzi dei server richiesti, ai quali inviare le richieste di chiamata.
- **SIP Registration Server**: mantiene un database delle registrazioni degli utenti. Questo elemento viene interrogato da un Proxy Server o da un Redirect Server per conoscere informazioni riguardo agli utenti chiamati. È utilizzato ad esempio per notificare all'infrastruttura SIP l'attuale locazione di un utente, così che si possa instradare la chiamata verso il terminale attualmente attivo.

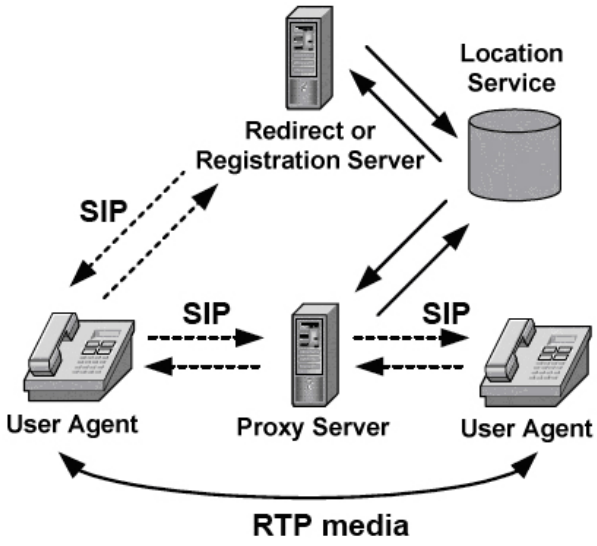

Figura 2.4 - Schematizzazione di un sistema SIP

La struttura degli indirizzi SIP è molto simile a quella degli indirizzi e-mail, infatti ogni utente SIP è identificato da un indirizzo gerarchico del tipo *sip:user@domain*.

2.3.2 Messaggi e Metodi

Quando parliamo di "invio di una chiamata" facciamo riferimento allo scambio di particolari messaggi tra l'utente chiamante ed il chiamato al fine di instaurare la comunicazione. In SIP i messaggi scambiati fra gli User Agent per l'instaurazione della chiamata, detti Request (messaggio inviato dallo UAC verso lo UAS) e Response (messaggio inviato dallo UAS verso lo UAC), sono di tipo testuale ed hanno dimensioni maggiori rispetto ai relativi messaggi H.323, ma sono di più facile comprensione e quindi di più semplice utilizzo per gli sviluppatori. Gli stessi tipi di messaggi sono utilizzati sia per la gestione della segnalazione che per la negoziazione della sessione tra due endpoint, sfruttando lo scambio di opportuni parametri codificati.

La sintassi dei messaggi in SIP è molto simile ad HTTP, a partire dal campo header atto a contenere informazioni fondamentali come il mittente ed il destinatario della chiamata, il tipo di codifica da adottare e le capacità supportate.

I messaggi tipicamente sono composti da:

- una riga iniziale;
- uno o più campi che compongono l'header;
- una riga vuota che indica la fine dell'header;
- una parte, opzionale, chiamata "message body".

```
generic-message = start-line
                  *message-header
                  CLRF
                  [message-body]

dove start-line = Request-Line / Status-Line
```

Figura 2.5 - Struttura del messaggio SIP

L'intestazione dei messaggi è differente per le Request e per le Response. Nelle Request si ha una Request-Line, mentre nelle Response si ha una Status-Line.

L'header è un insieme di attributi che definiscono i parametri della comunicazione, quali:

- *To*: indica l'indirizzo del destinatario in formato SipURI (Sip Uniform Resource Identifier);
- *From*: indica l'indirizzo SipURI del mittente;
- *Call-ID*: contiene un identificatore univoco per la sessione. È formato da una sequenza alfanumerica generata casualmente e dall'indirizzo IP del mittente;
- *CSeq*: serve per identificare e ordinare le transazioni;

- *Max-forward*: indica quante volte può essere ripetuto l'invio di un messaggio che non ha ancora ricevuto risposta (è decrementato ad ogni invio);
- *Via*: è l'indirizzo specificato dallo User Agent Client per ricevere i messaggi di tipo Response;
- *Contact*: indica l'indirizzo SipURI presso cui il mittente dei messaggi Request vuole essere contattato degli altri partecipanti alla sessione;
- *TransactionID*: è l'identificatore della transazione, univoco per la Request e la rispettiva Response;
- *Content-Type*: contiene una descrizione del body e del suo contenuto;
- *Content-Lenght*: indica la lunghezza del body.

Il body, quando presente, racchiude eventuali informazioni sul set di capabilities dello UA.

I messaggi SIP, come detto in precedenza, possono essere di due tipi: Request e Response. Vediamo le due tipologie nel dettaglio.

I **messaggi di tipo Request** sono inviati dagli UAC agli UAS. La riga iniziale, detta Request-Line, è composta dal nome del metodo richiamato, dalla Request-URI e della versione del protocollo SIP. La Request-URI è un SIP URL (Uniform Resource Locator) o un URI (Uniform Resource Identifier) generale, che permette di indicare l'utente o il servizio al quale si riferisce una particolare richiesta. Come detto in precedenza, un indirizzo SIP può essere descritto nella forma *user@host*, dove *user* è il nome dell'utente o il numero di telefono, *host* è il nome del dominio o l'indirizzo della rete.

I metodi definiti nella versione 2.0 del protocollo sono di quattro tipi: per la registrazione, per iniziare la sessione, per chiuderla e per richiedere informazioni sulle capabilities del server. I metodi principali, descritti in seguito, sono: INVITE, CANCEL, ACK, BYE, OPTIONS e REGISTER.

La Request di tipo INVITE è utilizzata dal chiamante (detto "callee") per invitare il chiamato (detto "called") a partecipare ad una conferenza. Il caso più semplice è una conferenza con solo due partecipanti, perciò in realtà è una comunicazione a due, ovvero una tradizionale telefonata VoIP. L'applicazione del

caller deve completare tutti i campi del messaggio da inviare, inoltre deve fornire anche informazioni sulle capabilities con cui si svolgerà la comunicazione, ovvero la descrizione delle potenzialità del sistema utilizzato dal caller per creare lo streaming audio e video da usare durante la conferenza.

```
INVITE sip:alice@127.0.0.7:5070 SIP/2.0
Via: SIP/2.0/UDP 127.0.0.8:5080
     branch=z9hG4bK776asdhds
To: <sip:alice@127.0.0.7:5070>
From: Peter <sip:peter@127.0.0.8:5080; transport=udp>;
      tag=3f71033f
Call-ID: 5a9ae778693cf64b@127.0.0.8
Contact: <sip:peter@127.0.0.8:5080>;transport=udp >
Max-Forwards: 70
CSeq: 1 INVITE
Content-Type: application/sdp
Content-Length: 140

v=0
o=Peter <sip:peter@127.0.0.8:5080>
s=Session SIP/SDP
c=IN IP4 127.0.0.8
```

Figura 2.6 - Esempio di Request INVITE

In figura 2.6 è presentata una tipica INVITE Request. La start-line definisce il metodo contenuto nel messaggio ed è seguita dai vari campi che compongono il messaggio. Da notare che i campi *Via*, *From* e *Contact* contengono lo stesso indirizzo SipURI: questo significa che chi ha inviato la Request desidera ricevere le

risposte allo stesso indirizzo e che anche le future Request del callee dovranno essere inoltrate sempre allo stesso recapito. Dai campi *Content-Type* e *Content-Lenght* si acquisiscono informazioni sul messaggio trasportato: nell'esempio di figura 2.6 si specifica una stringa esplicitata nel formato SDP e di lunghezza 140. La stringa SDP inclusa nel messaggio è la proposta fatta dal mittente al destinatario sul formato di trasmissione dei dati.

Come sarà discusso più avanti nella descrizione dei messaggi di tipo Response, in caso positivo (chiamata accettata), l'INVITE è seguito da una messaggio di tipo OK, mentre in caso negativo (chiamata rifiutata) è seguito da un messaggio di redirezionamento (es. MOVED TEMPORARILY), di errore del client (es. BUSY HERE), o di errore del server. La transazione dell'INVITE si considera invece terminata quando lo UAC invia allo UAS un messaggio ACK.

L'invio di un messaggio CANCEL avviene invece quando il caller, che ha già inoltrato un INVITE ma che è ancora in attesa di una risposta definitiva da parte del callee, decide di non voler più sostenere la comunicazione. Il CANCEL può essere invocato anche se lo UAC ha già ricevuto dei messaggi di tipo TRYING o RINGING, in quanto queste risposte non modificano lo stato della comunicazione, che può essere stabilita solo in seguito alla ricezione di un messaggio OK.

Differentemente dalla Request di tipo INVITE, nella Request di tipo CANCEL si nota la mancanza del campo *Content-Type*, dal momento che l'interruzione della comunicazione non richiede l'invio di informazioni aggiuntive.

Il metodo ACK è necessario per completare l'instaurazione di un collegamento SIP. Quando il callee riceve il messaggio ACK significa che può iniziare a ricevere e a trasmettere lo stream audio/video. La ricezione dell'ACK significa anche che il caller ha accettato le specifiche inoltrate dal callee su come deve avvenire la trasmissione delle informazioni.

La Request di tipo BYE è quella che permette di terminare una conferenza da parte di uno partecipanti.

La Request di tipo OPTIONS serve agli UA per chiedere informazioni sui capabilities set e può essere inoltrata verso un altro UA o verso un Proxy Server.

Infine la Request di tipo REGISTER è il messaggio con cui lo User Agent (in questo caso client) registra il proprio SipURI presso un server SIP.

I **messaggi di tipo Response**, differentemente dalle Request, presentano, come riga iniziale, una Status-Line, che consiste in una stringa contenente la versione del protocollo, lo Status-Code ed il relativo significato testuale.

Lo Status-Code è un codice numerico di tre cifre nella forma XYZ, in cui la cifra X definisce la tipologia parziale del messaggio. I codici usati sono molteplici, più nel dettaglio si hanno:

- 1YZ = risposta provvisoria, la Request è stata ricevuta, ma la risposta è ancora in fase di elaborazione:
  - 100 TRYING, il server sta cercando l'utente;
  - 180 RINGING, l'utente è stato localizzato e si è in attesa che risponda;
  - 181 CALL IS BEING FORWARDED, la chiamata è stata inoltrata ad un differente destinatario;
  - 182 QUEUED, il callee è momentaneamente non disponibile, ma la richiesta, anziché essere rifiutata, è stata messa in attesa;
  - 183 SESSION PROGRESS, questa risposta è utilizzata per portare informazioni sul progredire della sessione che non è stata altrimenti classificata.
- 2YZ = successo, la Request è stata correttamente ricevuta ed accettata:
  - 200 OK
- 3YZ = redirezionamento, devono essere intraprese altre azioni per poter completare la Request:
  - 300 MULTIPLE CHOICE, l'indirizzo specificato nella Request corrisponde a diverse scelte, ognuna delle quali ha la sua specifica ubicazione e lo UA può selezionare la comunicazione preferita e reinoltrare la sua richiesta a quella locazione;
  - 301 MOVED PERMANENTLY, l'utente non può più essere trovato all'indirizzo della Request-URI e lo UAC dovrebbe riprovare al nuovo indirizzo dato dal campo Contact dell'header;

  - 302 MOVED TEMPORARILY, come il precedente, con la differenza che la mancanza è momentanea.
- 4YZ = errore del client, la richiesta contiene degli errori o non può essere inoltrata al server:
  - 400 BAD REQUEST, la richiesta non può essere compresa a causa di un errore di sintassi;
  - 401 UNAUTHORIZED, la richiesta necessita di un'autenticazione dell'utente;
  - 408 REQUEST TIMEOUT, il server non riesce a rispondere in un tempo ragionevole;
  - 482 LOOP DETECT, il server ha riscontrato un ciclo infinito;
  - 486 BUSY HERE, il chiamato ha ricevuto con successo la richiesta, ma in questo momento non vuole o non è in grado di accettare chiamate aggiuntive;
  - 487 REQUEST TERMINATED, la richiesta è stata terminata da un BYE o da un CANCEL.
- 5YZ = errore del server, il server è impossibilitato a completare una richiesta apparentemente corretta:
  - 500 SERVER INTERNAL FAILURE, il server ha incontrato una condizione inaspettata che gli impedisce di completare la richiesta.
- 6YZ = fallimento globale, la richiesta non può essere completata da nessun server:
  - 600 BUSY EVERYWHERE, il chiamato è stato contattato con successo, ma è impegnato e non vuole accettare la chiamata. Questa risposta è data solo se il client sa che nessun altro terminale può essere contattato.

Vediamo più nel dettaglio i messaggi di Response usati più di frequente: OK, TRYING, RINGING, REQUEST TIMEOUT e BUSY HERE.

La Response OK, come già specificato, indica il successo dell'operazione. È usata in molteplici situazioni, ad esempio per confermare l'avvenuta ricezione di un messaggio oppure per accettare un INVITE. Nel caso sia spedita in risposta ad

un INVITE (v. figura 2.7), la Response OK contiene tutte le informazioni necessarie per avviare la comunicazione, come la descrizione delle capabilities supportate dallo UAS per lo scambio dei contenuti audio/video.

```
SIP/2.0 200 OK

Via: SIP/2.0/UDP 127.0.0.8:5080

To: <sip:alice@127.0.0.7:5070>;tag=8a7ba4f4

From: "Peter" <sip:peter@whiterabbit.com:5080>;tag=3f71033f

Call-ID: 5a9ae778693cf64b@127.0.0.8

CSeq: 1 INVITE

Contact: sip:alice@127.0.0.7:5070

Max-Forward: 70

Content-Type: application/sdp

Content-Length: 140


v=0

o="Peter" <sip:peter@ whiterabbit.com:5080>

s=Session SIP/SDP

c=IN IP4 127.0.0.7
```

Figura 2.7 - Esempio di Response OK

La Response TRYING indica che la Request è stata ricevuta generalmente da un intermediario come un Proxy Server, ma si è in attesa che sia elaborata.

La Response RINGING è spedita dallo User Agent Server che ha ricevuto l'INVITE. In questo caso sono già state accettate le specifiche della trasmissione, si è solo in attesa che il destinatario della chiamata decida se accettarla oppure no.

La Response REQUEST TIMEOUT è un messaggio inoltrato nel caso in cui la Request risulti pendente per troppo tempo, superando i tempi di attesa indicati. Non è specificato il motivo per cui si è verificato questo ritardo, potrebbe essere dovuto ad un indirizzo sbagliato o ad altro.

Infine il BUSY HERE è un messaggio definitivo di rifiuto. Indica che la Request inviata non è stata accettata e viene fornita la motivazione del rifiuto.

2.3.3 Comunicazione SIP

In questa sezione viene descritta nel dettaglio la successione logica dei messaggi Request/Response per l'instaurazione di una comunicazione SIP fra due User Agent. L'analisi consiste nella descrizione dell'ipotetica situazione mostrata graficamente in figura 2.8.

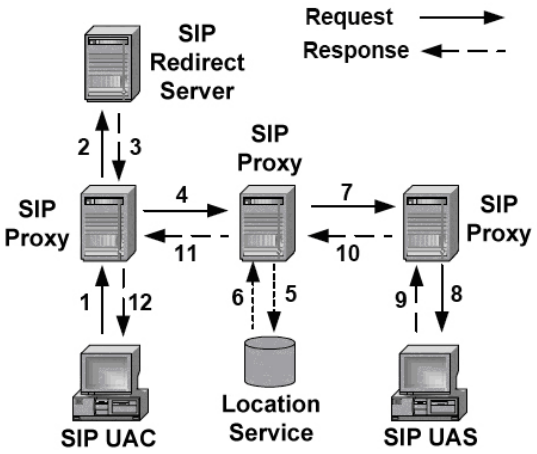

Figura 2.8 - Instaurazione di una comunicazione SIP

Il SIP User Agent chiamante crea una Request INVITE per l'utente SIP *sip:john@company.com*. La richiesta è inviata ad un Proxy Server locale (1), che la inoltra ad un Redirect Server (2), che gestisce tutte le SIP Request sul dominio. Il server relativo al dominio *company.com* riconosce l'utente *john*, ma questo utente è attualmente loggato come *john.user@university.edu*. Il server invia in risposta l'informazione di location dell'utente (3). Il Proxy Server ottiene via DNS

l'indirizzo IP del Proxy Server attivo sul dominio *university.edu*, sulla base di questa informazione la richiesta viene inoltrata (4). Il Proxy Server consulta il database locale (5), tramite il quale viene a conoscenza (6) che l'utente *john.user@university.edu* è identificato localmente all'interno del dominio come *j.green@tlc.university.edu*. Il server principale dei Proxy Server del dominio inoltra la richiesta al Proxy Server del dipartimento *Tlc*, che è a conoscenza dell'indirizzo IP locale dell'utente finale ed è in grado di inoltrargli la richiesta di chiamata.

Analizziamo ora nel dettaglio i messaggi scambiati tra le varie entità SIP: lo scenario più semplice possibile, descritto graficamente in figura 2.9, vede l'instaurazione di una chiamata SIP fra due device SIP operanti in modo diretto. Supponiamo ad esempio che entrambi gli utenti facciano parte di una medesima rete e conoscano reciprocamente gli indirizzi IP.

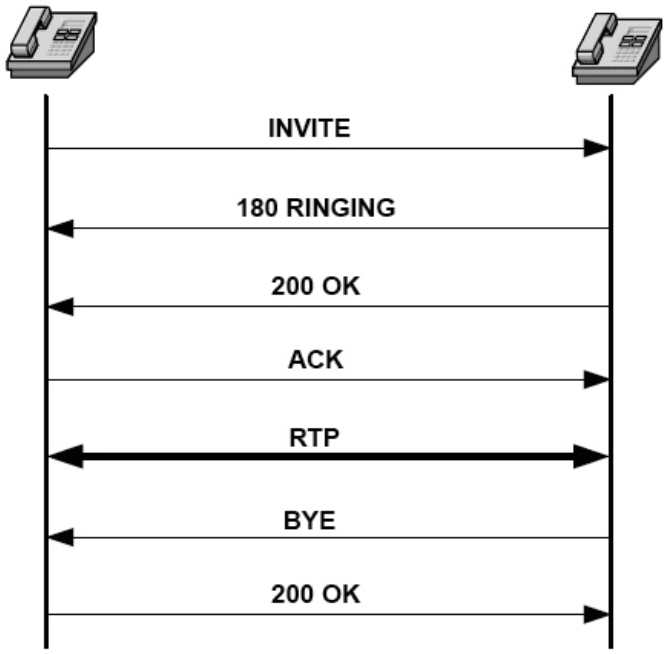

Figura 2.9 - Chiamata SIP diretta

L'utente chiamante (UAC) inizia la chiamata inviando un messaggio INVITE all'utente destinatario (UAS), che risponde con un messaggio 180 RINGING tramite il quale lo UAC viene informato che la chiamata ha raggiunto l'utente remoto. Quando l'utente chiamato risponde, viene inviato il messaggio 200 OK e, successivamente, il messaggio ACK sancisce l'avvenuto collegamento. Dopo lo scambio di dati basato su RTP, la fine della chiamata viene sancita dal messaggio BYE (confermato dal messaggio 200 OK).

Le due seguenti figure illustrano invece la successione di messaggi SIP in due situazioni più complesse: una chiamata SIP in presenza di un Proxy Server (figura 2.10) ed in presenza di un Redirect Server (figura 2.11).

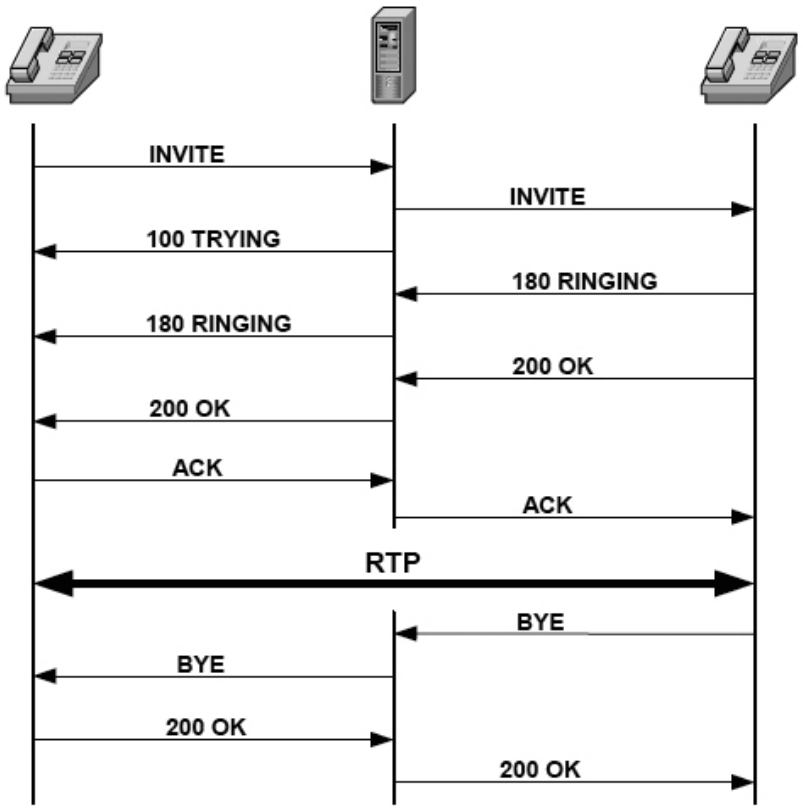

Figura 2.10 - Chiamata SIP con Proxy Server

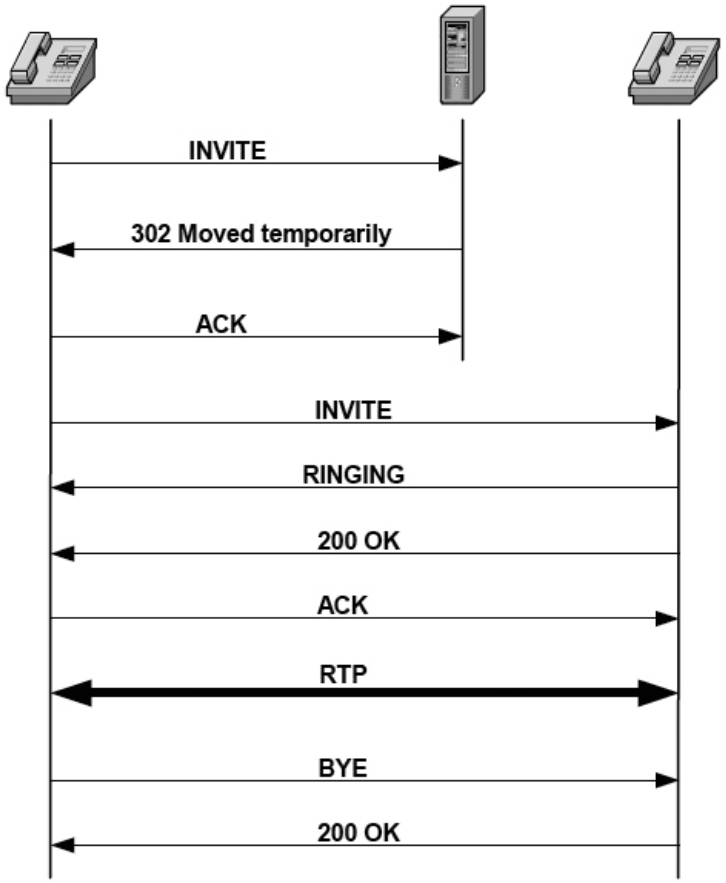

Figura 2.11- Chiamata SIP con Redirect Server

Vediamo infine un ultimo componente SIP, del quale si è già parlato nel corso della trattazione dello standard H.323 e che assume un ruolo fondamentale in termini di interoperabilità del protocollo: il **gateway**.

Un gateway SIP è un'applicazione che connette una rete di utenti SIP ad un'altra rete basata su un altro protocollo di segnalazione. Si potrebbe vedere il gateway come un particolare tipo di User Agent in grado di interagire con un altro protocollo. Un gateway si occupa tipicamente sia della conversione dei messaggi di segnalazione sia dei media veri e propri, ma può accadere che la sua funzione si

limiti solamente alla segnalazione. Questo è il caso, ad esempio, del gateway SIP-H.323 che si occupa della conversione dei messaggi di segnalazione tra SIP e H.323 senza influire sullo scambio di informazioni multimediali basato su RTP, che avviene in modo diretto fra i due terminali. Nel caso di una rete PSTN (Public Switched Telephony Network) invece il gateway si occupa della conversione sia della segnalazione sia dei media.

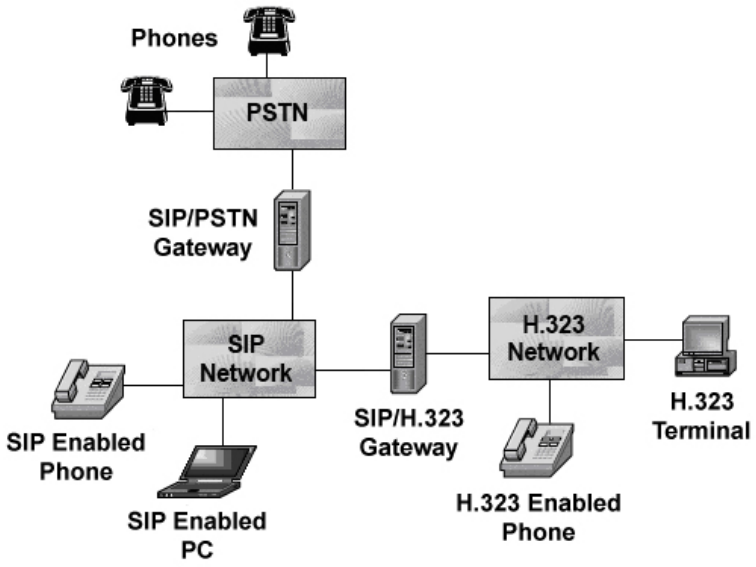

Figura 2.12 - Rete SIP con gateway

2.4 Analisi Comparata di H.323 e SIP

Nelle precedenti sezioni sono state descritte le caratteristiche principali dei protocolli H.323 e SIP. A questo punto viene presentata un'analisi comparata dei due standard che consentirà di capire a fondo quali sono gli elementi chiave che hanno fatto attualmente di SIP lo standard più utilizzato nel campo della tecnologia VoIP.

L'analisi si struttura su diversi campi: si parte valutando la filosofia alla base dello sviluppo dei due protocolli, successivamente vengono analizzati la complessità e le prestazioni (in termini di carico sulla rete), i metodi di codifica utilizzati per trasmettere i pacchetti, ed infine l'aspetto della scalabilità, che rappresenta senza dubbio un aspetto fondamentale nel momento in cui l'utenza cresce, sia dal punto di vista del numero di terminali, sia dal punto di vista tecnologico (ad es. con l'introduzione di nuovi codec).

2.4.1 Filosofie di Base

L'analisi delle filosofie che stanno alla base dello sviluppo dei due standard mette in luce una prima fondamentale differenza in termini di approccio.

H.323 adotta un approccio di tipo top-down, in cui lo schema concettuale viene prodotto a partire da uno schema iniziale mediante una serie di raffinamenti successivi. Questo schema di partenza descrive tutte le specifiche con pochi concetti astratti e viene poi affinato mediante opportune trasformazioni che aumentano il livello di dettaglio della rappresentazione. Lo standard H.323 necessita di un livello di dettaglio molto elevato in quanto il suo scopo è quello di descrivere una struttura di comunicazione completa: dai protocolli, alle entità che comunicano, alla serie di messaggi che rendono possibile lo scambio dei dati multimediali, alle soluzioni specifiche per garantire QoS e sicurezza.

SIP segue la filosofia della IETF, secondo la quale i sistemi e le applicazioni dovrebbero essere creati utilizzando moduli il più possibile generici. Tutti i protocolli standardizzati dalla IETF sono indipendenti dallo specifico sistema o dalla specifica applicazione in cui verranno utilizzati, per questo motivo SIP non dispone ad esempio di soluzioni per garantire QoS. Possiamo sostanzialmente affermare che l'approccio di SIP, a differenza di H.323, segua una sorta di logica bottom-up, in cui le specifiche iniziali sono suddivise in componenti via via sempre più piccole, fino a descrivere piccoli frammenti del problema. Le componenti vengono poi fuse con trasformazioni successive, per giungere allo schema concettuale finale.

2.4.2 Complessità

Dalle descrizioni dei due standard presentate nelle sezioni precedenti emerge senza dubbio una maggiore complessità di H.323 rispetto a SIP. Analizzando nello specifico i documenti che definiscono i due standard, oltre a notare un volume di informazioni e di specifiche di gran lunga superiore nel caso di H.323, è possibile capire rapidamente che H.323, a differenza di SIP, raggiunge un altissimo livello di dettaglio nella descrizione di ogni componente e di ogni protocollo che comprende. Inoltre, l'elevata complessità di H.323 è dovuta alla presenza di una vasta gamma di protocolli diversi, il cui utilizzo molte volte non è esclusivo: basti pensare alla fase di instaurazione della chiamata, nella quale entrano in gioco ben tre diversi protocolli (Q.931, H.245 e H.225).

Anche confrontando la successione di messaggi necessaria per creare una comunicazione fra due terminali, la complessità di H.323 risulta sicuramente maggiore: il numero di messaggi scambiati nel caso di SIP è circa quattro volte inferiore rispetto ad H.323.

Per quanto riguarda il trasporto dei media, H.323 inizialmente supportava solo TCP, solo dalla terza versione del protocollo è stato aggiunto il supporto per UDP. TCP permette di ottenere una connessione sicura ed affidabile, ma di contro comporta un aumento non indifferente del carico sulla rete. Si deve aggiungere che il messaggio Q.931 utilizzato da H.323 per aprire il primo canale di comunicazione è spesso più grande della MTU (Maximum Transmission Unit) della rete e quindi richiede almeno due pacchetti TCP, il secondo dei quali deve attendere un messaggio di ACK da parte del destinatario prima di essere inviato, con un conseguente aumento del ritardo di propagazione (round-trip delay).

A differenza di H.323, SIP, fin dalla prima versione, è in grado di supportare sia UDP che TCP: si è visto che in SIP lo scambio di messaggi si basa su una logica Request/Response simile ad HTTP e come in HTTP i dati sono successivamente incapsulati all'interno di datagrammi UDP.

I pacchetti TCP sono più complessi rispetto a quelli UDP, la maggiore complessità è data da un maggior numero di field necessari per garantire la qualità

del servizio. L'aggiunta del supporto UDP per la terza versione di H.323 ha semplificato notevolmente il processo di instaurazione di chiamata, che ora è molto simile tra i due protocolli.

Anche la fase preliminare alla comunicazione atta a stabilire le capacità multimediali supportate dai terminali risulta essere più complessa in H.323: prima che la comunicazione possa iniziare, è necessario che gli endpoint si scambino tutte le informazioni sui propri supporti multimediali e decidano chi rivestirà il ruolo di master e chi il ruolo di slave durante la chiamata. La struttura di SIP è molto più semplice: il terminale UAC incorpora nel messaggio Request INVITE un message body SDP in cui sono contenute tutte le capacità supportate, mentre il terminale UAS le inserisce in un message body SDP contenuto all'interno del messaggio Response 200 OK. Inoltre, data la natura client-server di SIP, non è necessario nominare un master e uno slave: chi invia la richiesta di chiamata riveste il ruolo di client (UAC), chi la riceve opera da server (UAS).

2.4.3 Codifica dei Dati Trasmessi

Un'altra importante differenza tra i due protocolli consiste nella codifica utilizzata nella trasmissione dei dati.

H.323 utilizza una codifica binaria ASN.1+BER o ASN.1+PER. ASN.1 (Abstract Syntax Notation One) è il linguaggio per la definizione degli standard, indipendente dall'implementazione. Si deve fare attenzione sul vero scopo di ASN.1, che non è riferito ad uno standard, ad un metodo di codifica, ad un linguaggio di programmazione o ad una specifica piattaforma: ASN.1 è il linguaggio degli standard, cioè uno standard è scritto in ASN.1. BER (Basic Encoding Rules) definisce un insieme di regole per codificare i dati su ASN.1. Esistono altri metodi di codifica di ASN.1 tra i quali: PER (Packed Encoding Rules), CER (Canonical Encoding Rules) e DER (Distinguish Encoding Rules). Per avere ben chiara la distinzione possiamo paragonare ASN.1 ad un linguaggio di programmazione e PER al compilatore per quel linguaggio.

SIP si basa invece su ABNF (Augmented Backus-Naur Form), una versione modificata della precedente BNF. Lo scopo di ABNF è praticamente lo stesso di

ASN.1, cioè si tratta un linguaggio per definire il formato e la sintassi delle specifiche, ma, diversamente da ASN.1, ABNF utilizza una codifica testuale che ricorda da vicino HTTP o XML.

Possiamo affermare che una codifica binaria abbia una maggiore efficienza rispetto ad una equivalente ma testuale. Tuttavia il successo di molte applicazioni Internet basate su codifiche testuali deve far dedurre che l'efficienza non rappresenta un aspetto critico per lo sviluppo. Bisogna inoltre aggiungere, a favore della codifica testuale, che quando si riceve o si invia un messaggio H.323 con una codifica binaria è necessario un dispositivo encoder/decoder in ogni terminale H.323, che comporta un inevitabile aumento di complessità.

2.4.4 Scalabilità

La scalabilità è probabilmente l'aspetto più significativo per il confronto tra H.323 e SIP, in quanto rappresenta la possibilità di evoluzione del protocollo per potersi affermare e diventare uno standard perdurante nel tempo.

Anche in termini di scalabilità SIP offre sicuramente garanzie migliori rispetto ad H.323. Consideriamo ad esempio la possibilità di introdurre di nuovi header all'interno di un messaggio gestito da SIP per supportare nuove funzionalità: ogni utente ha la possibilità di inserire un nuovo header, a condizione che questo venga precedentemente registrato presso IANA (Internet Assigned Number Authority). Tuttavia anche l'assenza di questa registrazione non comporta degli errori nella comunicazione in quanto se uno User Agent SIP riceve un messaggio contenente degli header di cui non conosce il significato, si limita ad ignorarli.

Nel caso di H.323 il procedimento è più complicato. La codifica ASN.1 consente la definizione di nuovi header attraverso l'utilizzo di un particolare campo (*nonStandardParam*). Il nuovo parametro definito conterrà un codice identificativo ed un valore numerico che rimane oscurato e quindi ha validità solo per quel particolare utente. Esistono tuttavia diverse limitazioni legate soprattutto al poco spazio a disposizione dei campi *nonStandardParam*.

Un altro punto a svantaggio della scalabilità di una rete H.323 è legato alla retrocompatibilità: per far sì che terminali che utilizzano differenti versioni dello

standard H.323 possano comunicare, è necessario che qualsiasi elemento della versione obsoleta venga mantenuto in una successiva con un conseguente aumento del numero di dati che devono essere codificati e scambiati. SIP invece tende lentamente ad eliminare quegli header che con il tempo sono divenuti obsoleti e in questo modo il protocollo viene mantenuto leggero, preciso ed ordinato.

Un aspetto da considerare nella valutazione della scalabilità dei due standard VoIP riguarda la capacità di gestione dei codec audio/video. Nel caso di H.323, un codec prima di essere utilizzato deve essere registrato e standardizzato: il problema è che non tutti gli utenti hanno la possibilità di farlo, anzi solo la ITU-T può definire nuovi codec. In SIP invece un endpoint utilizza il protocollo SDP per informare l'endpoint chiamato su quali codec è in grado di supportare, ed essendo i codec rappresentati solamente da un nome che chiunque può registrare presso IANA, SIP è sostanzialmente in grado di supportare tutti i codec audio e video disponibili in circolazione.

2.4.5 Analisi delle Prestazioni e Conclusioni

Dopo aver analizzato in modo comparato H.323 e SIP da un punto di vista teorico, sono stati effettuati alcuni test pratici per misurare praticamente l'efficienza dei due standard.

La fase di testing si è basata sull'utilizzo di alcune applicazioni open-source basate su H.323 e SIP. In particolare, nel caso di H.323 le analisi prestazionali hanno coinvolto i terminali OhPhone, OpenAM ed il gatekeeper OpenH323 Gatekeeper [41], mentre per la valutazione di SIP è stato usato il client SIP Communicator [45] ed il SIP Proxy Server remoto Iptel.org [23]. Per l'analisi del traffico di rete è stato utilizzato Ethereal [13], un software freeware che permette l'analisi dei dati scambiati tra i terminali coinvolti nelle comunicazioni.

Tramite la serie di test effettuati è stato possibile ottenere una conferma pratica delle valutazioni che erano emerse dall'analisi teorica descritta nelle sezioni precedenti: anche a livello pratico e prestazionale SIP si è dimostrato decisamente più versatile e leggero rispetto ad H.323.

In conclusione possiamo affermare che H.323, pur funzionando correttamente e non sovraccaricando eccessivamente la rete, comporta un carico decisamente maggiore rispetto a SIP, che presenta una struttura molto leggera sia in fase di instaurazione che di chiusura di chiamata. Inoltre il protocollo SIP offre maggiori garanzie da un punto di vista della scalabilità e quindi in vista di possibili future modifiche ed evoluzioni: SIP, seguendo la filosofia dei suoi sviluppatori, è in grado di aggiornarsi e di supportare tutti i codec audio/video in circolazione, processo che risulta più macchinoso e non sempre possibile nel caso di H.323. L'unico punto a favore di H.323 è rappresentato dalla codifica utilizzata: la sua codifica binaria richiede l'invio di un numero minore di ottetti e questo fa sì che i pacchetti di H.323 siano più leggeri (in termini di Kbytes) rispetto a quelli SIP, che utilizza invece una codifica testuale simile ad XML o HTTP.

2.5 Sviluppo di una MCU SIP

Una MCU (Multipoint Control Unit), come già accennato nel corso della trattazione dello standard H.323, è un terminale di un sistema di comunicazione per conferenze audio/video. La sua funzione principale è quella di intercettare il traffico di dati proveniente da ogni partecipante della conferenza ed inoltrarlo verso tutti gli altri, ma è anche possibile che svolga funzioni di controllo, gestione e supervisione della conferenza.

Una delle prime fasi di sviluppo software sostenute durante l'attività di ricerca sul VoIP ha riguardato l'implementazione di una MCU per SIP a partire dal SIP client open-source SIP Communicator [45], già citato in precedenza. Essendo basata su un client SIP classico, la MCU appare alla rete come se fosse un qualsiasi terminale che si registra normalmente ad un server SIP ottenendo un numero SIP a lui associato. Tutti gli utenti chiamano questo numero e una volta avviata la conferenza vengono settate delle connessioni punto-punto tra ogni partecipante e la MCU. Il traffico entrante viene quindi mixato e trasmesso in uscita in broadcast.

Tale tipo di soluzione presenta i seguenti vantaggi:

- il comportamento esterno della struttura implementata è del tutto simile a quella di un qualsiasi altro client, per cui tale soluzione non comporta problemi di alcun tipo nell'interfacciamento con il resto del dominio SIP;
- la struttura realizzata è di tipo periferico per cui qualunque client potrebbe in teoria avviare una conferenza a prescindere dai servizi offerti dal server che si sta utilizzando.

2.5.1 Software e Librerie Utilizzati

Come già accennato, la MCU è stata realizzata a partire dal client SIP Communicator, un software open-source scritto in Java basato sulle librerie NIST-SIP (che implementano le JAIN-SIP API e le JAIN-SDP API) per la parte concernente al livello SIP e SDP e sulle librerie JMF (Java Media Framework) per quanto riguarda la gestione dei media e delle periferiche (v. figura 2.13).

JMF è un'API (Application Programming Interface) per la gestione di dati audio/video all'interno di applet ed applicazioni Java. JMF permette tra l'altro di riprodurre, catturare e salvare flussi multimediali, di convertire e gestire diversi formati di compressione audio/video e si occupa inoltre della gestione dei flussi RTP.

Per la fase di testing presso il dipartimento è stato utilizzato VOCAL (Vovida Open Communication Application Library) [48], un sistema distribuito di server che operano su macchina UNIX e forniscono servizi telefonici VoIP. VOCAL supporta periferiche che comunicano utilizzando il protocollo SIP come protocollo applicativo. La necessità di utilizzare un server interno deriva dal fatto che la rete del dipartimento è una LAN interfacciata ad Internet tramite un NAT (Network Address Translation) che traduce gli indirizzi locali delle intestazioni dei pacchetti uscenti dalla rete in indirizzi pubblici. SIP, così come H.323, invia esplicitamente gli indirizzi IP e le porte UDP/TCP anche nella parte dati dei messaggi TCP e tale parte non viene gestita dal NAT. Se tali indirizzi vengono usati da host esterni, questi ultimi non riuscirebbero ad instradare correttamente i pacchetti. In questi casi sono necessari degli apparati ALG (Application Level Gateway), ovvero

funzioni implementate nel nodo NAT che elaborano i pacchetti analizzando e modificando il contenuto dati di livello applicativo.

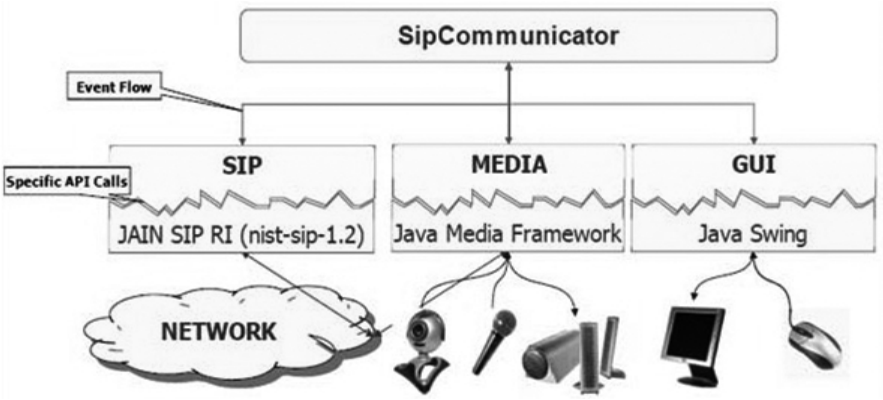

Figura 2.13 - SIP Communicator

2.5.2 Gestione di Flussi RTP con JMF

In questa sezione si illustrano brevemente i passi principali per la creazione e la gestione dei flussi RTP e le principali classi e metodi utilizzati.

Con JMF (Java Media Framework), per catturare dati multimediali da periferiche di input quali un microfono o una videocamera, vengono solitamente seguiti i seguenti passi:

- viene localizzata la periferica di cattura da un *CaptureDeviceManager*;
- viene ricavato un oggetto *CaptureDeviceInfo* per la periferica;
- si ottiene un *MediaLocator* dall'oggetto *CaptureDeviceInfo* e lo si usa per creare un oggetto *DataSource*;
- viene creato un *Processor* utilizzando il *DataSource*;
- si avvia il *Processor* per iniziare il processo di cattura.

Prima di avviare la trasmissione si configura il *Processor* per generare il flusso RTP data nel formato desiderato. Se le tracce audio/video del *Processor* sono di diverso formato, ogni flusso multimediale viene trasmesso in una sessione RTP separata.

Una volta settati i formati delle tracce del *Processor* è possibile ottenere l'output come *DataSource*, che può essere un *PushBufferDataSource* oppure un *PullBufferDataSource* a seconda della sorgente dei dati. Il *DataSource* viene infine collegato al *SessionManager* (nel nostro caso particolare ad un *RTPManager*). Se il *DataSource* contiene *SourceStreams* multiple, ogni *SourceStream* viene trasmesso in flussi RTP separati.

Per quanto riguarda la ricezione bisogna impostare un *RTPManager* come ascoltatore di eventi *NewReceiveStreamEvent*. Quando viene rilevato un evento *NewReceiveStreamEvent*, viene ricavato il *ReceiveStream* e quindi il *DataSource*. Il risultato di tale metodo viene passato al *Manager* per costruire un *Player* per riprodurre lo stream multimediale. Affinché tutto funzioni correttamente è necessaria la presenza di corretti plug-in per la decodifica dei dati RTP.

2.5.3 SIP Communicator

SIP Communicator [45] è un soft-phone SIP open-source scritto in Java da Emil Ivov che permette di gestire chiamate, partecipare a sessioni audio/video multicast, registrarsi ad un SIP Registrar Server e visualizzare i messaggi in ingresso.

Dal punto di vista tecnico, SIP Communicator viene gestito da tre classi manager: *GUIManager*, *MediaManager* e *SipManager*, che si occupano rispettivamente di tutto ciò che concerne l'interfaccia grafica, i media in ingresso e in uscita ed i messaggi SIP. Viene analizzata in dettaglio la parte concernente la gestione dei media, in quanto è la parte di SIP Communicator che è stata maggiormente modificata per l'implementazione della MCU.

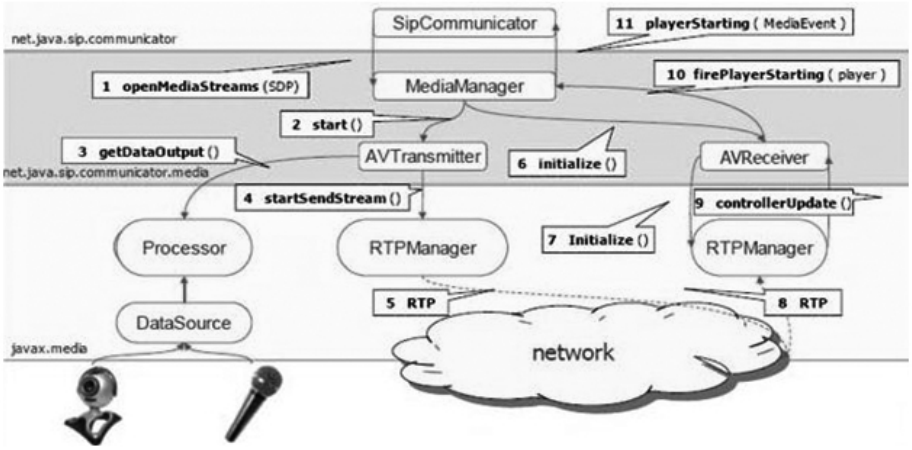

Figura 2.14 - Schema di funzionamento di SIP Communicator

Sulla base della schematizzazione mostrata in figura 2.14, una volta che la comunicazione viene stabilita, viene chiamato il metodo *openMediaStreams*, che si occupa in primo luogo di recuperare tutti i dati che riguardano la connessione e la sessione stabilita tramite un messaggio SDP. Le classi che si occupano principalmente della trasmissione e della ricezione sono *AVTransmitter* e *AVReceiver*.

Una volta avviato il trasmettitore viene configurato il *Processor*. Durante questa fase il *Processor* acquisisce le informazioni della sorgente necessarie per la sua programmazione. Dopo aver limitato i formati disponibili ai soli formati RTP validi, la costruzione del *Processor* viene completata e viene infine creato il *DataSource* di uscita. La trasmissione dei *DataSource* viene affidata a degli *RTPManager* (uno per ogni flusso in uscita). Si avvia infine il *Processor*.

All'avvio del ricevitore vengono aperti tanti *RTPManager* quante sono le sessioni attive. Ogni *RTPManager* aperto viene quindi posto come ascoltatore di tre tipi di eventi: *ReceiveStreamEvent*, *SendStreamEvent* e *SessionEvent*. Tra i vari tipi di eventi *ReceiveStreamEvent* gestiti riveste particolare importanza il *NewReceiveStreamEvent*, che informa l'ascoltatore che è stato rilevato un nuovo flusso di pacchetti RTP da una sorgente che non aveva mai trasmesso

precedentemente. Il SIP Communicator intercetta tali eventi, ricava il flusso RTP in questione e lo passa ad un *Player* per eseguirlo.

2.5.4 Implementazione e Testing

I punti principali su cui il comportamento di una MCU differisce da quello di un comune endpoint come SIP Communicator sono i seguenti:

- trasmissione verso più di un utente;
- gestione simultanea ed omogenea delle connessioni SIP verso tutti gli utenti della conferenza;
- possibilità di ricevere ma non inviare una chiamata;
- *DataSource* proveniente dal traffico in ingresso e non da periferiche di cattura.

Per quanto riguarda l'interfaccia grafica è stata eliminata il pannello "dial" che consentiva di chiamare altri client in quanto, come accennato, tale funzione non è di norma implementata in una MCU. Sono state inoltre modificate le azioni dei listener associate alla pressione dei tasti per accettare/rifiutare le chiamate, in maniera tale che l'azione eseguita non sia più eseguita verso il singolo utente selezionato nella contact list ma verso tutti gli utenti collegati in quel momento alla MCU. In tal modo, ad esempio, il tasto "accept" avvierà la comunicazione verso tutti gli utenti che stanno chiamando e, una volta premuto, verrà disabilitato finché la conferenza non ha termine.

Per poter trasmettere verso tutti gli utenti della conferenza vi era la necessità di ricavare e immagazzinare tutti gli indirizzi IP dei partecipanti per poi utilizzarli nella fase di configurazione del *Transmitter*. A tal fine è stato modificato l'handler chiamato in risposta alla pressione del tasto di "accept": l'indirizzo IP viene estrapolato ed inserito in un array statico, per essere poi utilizzato nel momento dell'inizializzazione del *Transmitter*.

L'ultima modifica sostanziale riguarda la gestione degli eventi *NewReceiveStream*. Nel SIP Communicator essi venivano infatti gestiti dall'*AVReceiver* mentre nell'implementazione della MCU tale funzione viene delegata agli *AVTransmitter*. Quando viene rilevato infatti un *NewReceiveStream*,

viene ricavato il *DataSource* associato al flusso rilevato ed aggiunto ad un array statico. Una volta che il numero di *DataSource* registrati eguaglia il numero di partecipanti alla conferenza, il *DataSource* ottenuto viene settato come *DataSource* da trasmettere.

Per la fase di testing dell'applicazione sviluppata sono state eseguite prove in diverse modalità. Come SIP UA è stato utilizzato il client freeware X-Lite su macchine Windows.

Una prima fase di testing ha riguardato il funzionamento della MCU all'interno della rete LAN del dipartimento utilizzando come server SIP il sistema VOCAL [48] (e quindi un server locale). I risultati sono stati diversi e contraddittori: si sono alternate infatti sessioni di testing in cui si è ottenuto complessivamente audio di buona qualità con basso delay ad altre in cui l'audio ricevuto è risultato discontinuo e leggermente distorto. Tali situazioni possono derivare da diversi cause: sovraccarico della macchina locale su cui è in azione il server SIP, situazioni di sovraccarico della rete o ancora problemi con il firewall locale che comportano una perdita di pacchetti.

La seconda fase di testing ha riguardato l'utilizzo di un server SIP remoto fornito da Iptel.org [23], già citato in precedenza. In un primo momento si sono riscontrati dei problemi nell'indirizzamento dei pacchetti UDP contenenti i messaggi di gestione delle connessioni SIP: ad esempio un messaggio di INVITE diretto ad un particolare utente veniva recapitato a tutti gli utenti SIP attivi in quel momento. In seguito si è scoperto che tale problema era causato dalla presenza del server SIP locale che evidentemente intercettava e deviava i messaggi SIP. Risolto tale problema si sono ottenuti risultati simili alla precedente fase di testing.

Un altro dato importante è che i risultati ottenuti risultano omogenei su test eseguiti consecutivamente in un arco di tempo contiguo. Come conseguenza di tale risultato è lecito pensare che i malfunzionamenti dell'applicazione siano legati a particolari situazioni della rete in determinati momenti. A sostegno di tale ipotesi vi sono anche i risultati ottenuti nelle terza ed ultima fase di testing, che ha riguardato l'utilizzo del sistema VOCAL all'interno di LAN locali formate con un hub a 10 Mbit o con switch a 100 Mbit. In entrambi i casi l'applicazione ha fornito buone

prestazione non mostrando malfunzionamenti o cattiva qualità audio. Tali risultati tenderebbero inoltre ad escludere errori architetturali della MCU.

2.6 Il Sistema JadeSip

Un'altra interessante attività di ricerca relativa alla tematica VoIP ha riguardato la progettazione e la realizzazione del sistema JadeSip. JadeSip, come si può dedurre dal nome stesso, consiste nell'integrazione di due architetture in un'unica applicazione in grado di fornire, da un lato, servizi di telefonia e messaggistica basati su SIP, e, dall'altro, una serie di servizi di assistenza agli utenti basati sull'implementazione di una piattaforma ad agenti JADE. L'introduzione della tecnologia ad agenti consente di gestire in modo agevole molte situazioni che si possono verificare nel corso di una sessione multimediale, dalla registrazione ad un servizio di presenza, alla ricerca di utenti conosciuti, alla configurazione di particolari politiche di ricezione delle chiamate nel caso l'utente non sia disponibile. In questo senso ad ogni utente del sistema (in modo simile ai sistemi RAP e CAFE descritti nel primo capitolo) viene associato un agente personale in grado di predisporre l'interfaccia SIP per elaborare i contenuti in ingresso in modo opportuno: ad esempio, se lo stato dell'utente è "occupato", l'agente potrebbe decidere di ignorare i messaggi in ingresso, o in alternativa registrare la chiamata inviando successivamente una notifica al proprio utente.

In questo modo si delinea un'architettura dove gli agenti comunicano tra di loro al di sopra del substrato SIP e, al contempo, possono modificare il modo con cui la loro interfaccia SIP riceve o invia dati multimediali.

A livello implementativo, la principale attenzione è stata rivolta verso l'integrazione dei due middleware che compongono l'architettura, giungendo allo sviluppo di un'applicazione client basata su una piattaforma JADE ed in grado di fornire servizi di telefonia SIP. Ogni client è quindi dotato di un proprio agente JADE che gli consente di svolgere tutte le funzioni opportune.

Successivamente lo sviluppo ha portato alla creazione di un servizio di presenza JADE. Questo consiste in un'applicazione, basata interamente su JADE, che

fornisce un servizio di registrazione per gli agenti che lo richiedono: un agente di tipo client quando diventa attivo si registra a questo server (detto anche "repository") e gli comunica il proprio stato. Questo servizio dà la possibilità agli agenti (e quindi agli utenti) attivi nel sistema SIP di verificare la presenza di eventuali utenti conosciuti per instaurare con loro una comunicazione.

2.6.1 Architettura del Sistema

Come accennato in precedenza, il sistema JadeSip è composto da una parte client con cui interagisce l'utente finale e da un servizio di presenza JADE che fornisce ai client servizi di registrazione e di ricerca.

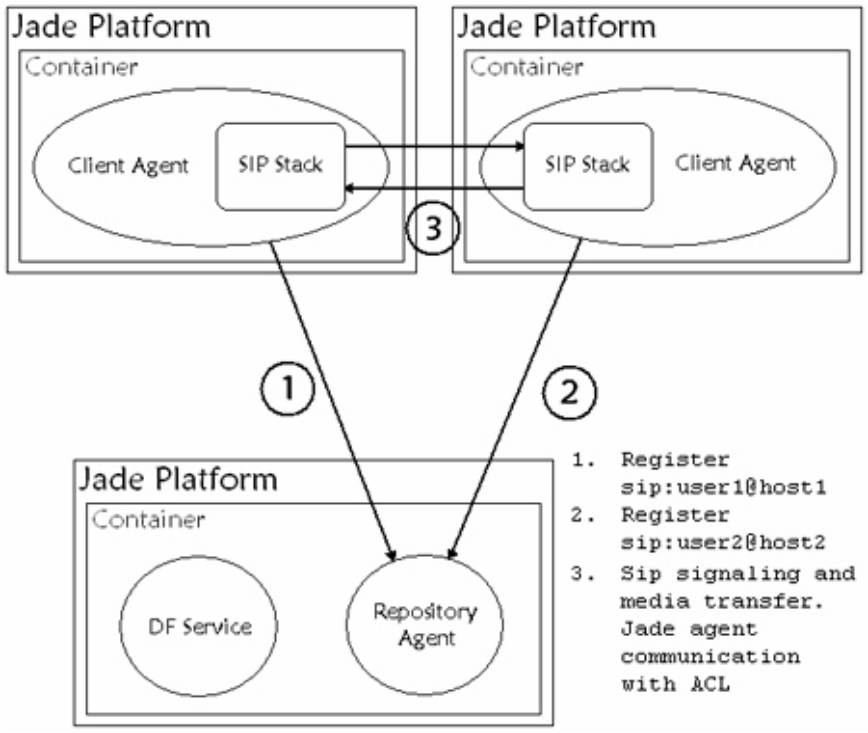

Figura 2.15 - Architettura del sistema JadeSip

In figura 2.15 è mostrata la successione logica delle operazioni necessarie per l'instaurazione di una chiamata fra due utenti del sistema JadeSip:

1) il Client Agent del primo utente si registra presso il server Repository Agent;
2) anche il Client Agent del secondo utente si registra presso il Repository Agent ed i due agenti vengono a conoscenza l'uno dell'altro;
3) ha inizio la comunicazione con l'instaurazione di un canale di comunicazione diretto tra i due client, tramite il quale vengono scambiati sia i messaggi SIP sia i messaggi ACL di iterazione fra gli agenti.

A livello di codice Java, il sistema si struttura in tre package principali: il package *client*, il package *server* ed il package *user*. Analizziamoli nel dettaglio.

L'**architettura del client** (**package *client***) comprende tre differenti aspetti, infatti l'applicazione deve provvedere alla creazione di un agente, di uno stack SIP, per l'implementazione del quale è stata utilizzata la libreria MjSip [40], ed ovviamente di una interfaccia grafica in grado di gestire l'interazione con l'utente. Per questo motivo, si è scelto di suddividere il package *client* in tre differenti sotto-package, come mostrato in figura 2.16.

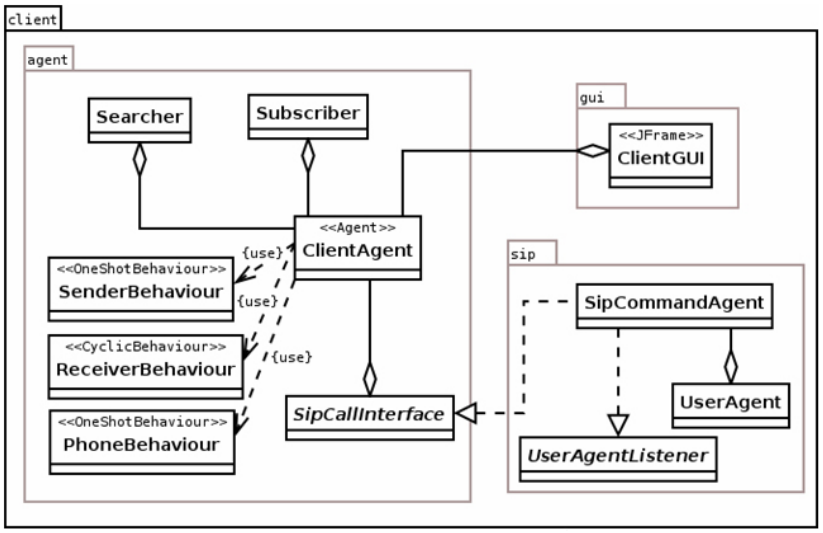

Figura 2.16 - Class diagram del package *client*

Il package *client.agent* si compone delle seguenti classi:

- *ClientAgent*: rappresenta il cuore dell'applicazione. Essa contiene un gran numero di altri oggetti, che vengono utilizzati per determinati scopi, ad esempio per abilitare la comunicazione e fornire il servizio di telefonia, per consentire lo scambio di messaggi ACL verso altri agenti, ecc.;
- *Searcher*: rappresenta una sorta di motore di ricerca, utilizzato sia per ricercare un repository esistente, sia per interrogare il repository stesso in modo da ottenere l'elenco degli utenti "amici" in linea;
- *Subscriber*: si occupa della gestione e dell'aggiornamento dello stato dell'utente;
- *SenderBehaviour*: rappresenta l'interfaccia di comunicazione verso gli altri agenti del mondo esterno;
- *ReceiverBehaviour*: è la classe duale alla classe *SenderBehaviour*. Viene lanciata durante la fase di setup dell'agente e non viene mai più rimossa. Questo consente di avere sempre in ascolto un behaviour per ogni tipo di messaggio in entrata, centralizzando dentro questa classe la gestione dei messaggi provenienti da altri agenti;
- *PhoneBehaviour*: contiene al suo interno le chiamate al protocollo SIP;
- *SipCallInterface* (interfaccia): contiene semplici metodi di base per poter effettuare una chiamata. Rappresenta il punto di interconnessione tra una qualunque implementazione del protocollo SIP e l'applicazione JADE realizzata. Dovrebbe risultare piuttosto agevole, in questo modo, passare ad un'altra implementazione SIP e sostituirla completamente (notare che l'implementazione attuale si trova nel package *client.sip*).

Il package *client.sip* si compone delle seguenti classi:

- *UserAgent*: si occupa della gestione delle chiamate ai metodi verso lo stack SIP, inizializzare l'audio e riprodurre gli eventi sonori del caso;
- *SipCommandAgent*: consente la chiamata SIP e gestisce gli eventi SIP lanciati dalla classe *UserAgent*;
- *UserAgentListener* (interfaccia): dichiara i metodi di callback chiamati dallo User Agent quando vengono ricevuti messaggi SIP di controllo.

Infine il package *client.gui* comprende solo la classe *ClientGUI*, che implementa l'interfaccia grafica dell'agente, consentendo all'utente di effettuare chiamate SIP, registrarsi al repository, cambiare il proprio stato ed annullare la registrazione. In figura 2.17 è mostrata una schermata della GUI.

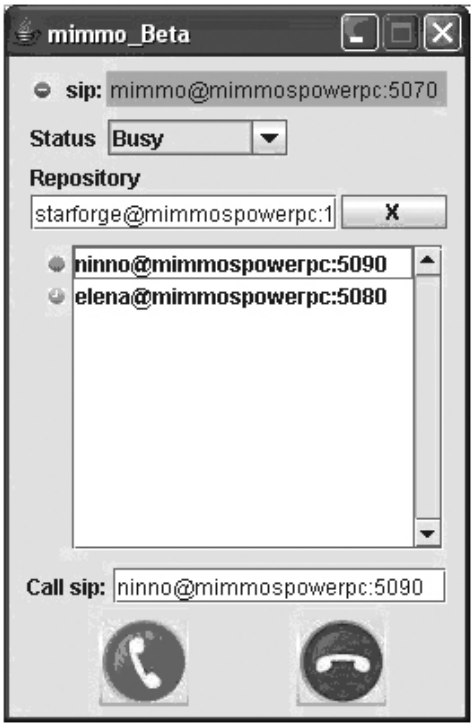

Figura 2.17 - GUI del sistema JadeSip

L'**architettura del server** (**package *server***) è probabilmente più semplice della precedente, in quanto il lato server del sistema non si deve occupare di interagire con diversi tipi di entità, ma deve curare solamente la comunicazione con agenti di tipo client. Lo schema dei package e delle classi contenute è di conseguenza più snello, come mostrato in figura 2.18.

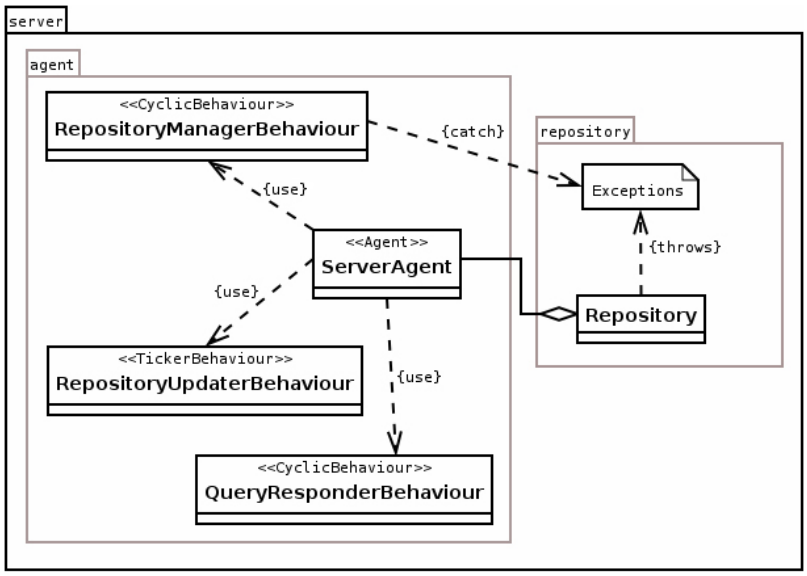

Figura 2.18 - Class diagram del package *server*

Il package *server.agent* si compone delle seguenti classi:

- *ServerAgent*: è il cuore dell'applicazione server in quanto implementa la gestione del repository e la comunicazione con gli utenti;
- *RepositoryManagerBehaviour*: si mette in ascolto dei messaggi ACL inviati dai client e si occupa di effettuare tre diverse operazioni: registrazione (aggiunta delle informazioni dell'utente nel repository), deregistrazione (rimozione dell'utente dal repository e notifica ai client interessati l'evento) ed aggiornamento (aggiornamento dello stato dell'utente);
- *RepositoryUpdaterBehaviour*: si occupa di effettuare un controllo periodico degli utenti in linea, in modo da verificare che qualcuno di questi non sia stato disconnesso per un qualche errore. Siccome la cancellazione della registrazione di un utente deve essere fatta esplicitamente, nel caso un utente esca in modo inaspettato (ad esempio viene a meno la connessione), è necessario predisporre un meccanismo per verificare periodicamente che

non rimangono registrati utenti "fantasma", che impedirebbero la registrazione di ulteriori utenti con lo stesso nome;

- *QueryResponderBehaviour*: il repository ha anche un'altra funzione, ovvero quella di accettare le richieste, da parte di utenti, di verificare la presenza dei loro utenti amici. Questa classe risponde a questa esigenza, fornendo informazioni dettagliate sugli utenti desiderati.

Infine il package *server.repository* comprende solo la classe *Repository*, atta a contenere le informazioni relative agli utenti. Le informazioni sono rappresentate dalle classe *User* descritta in seguito.

Come accennato in precedenza, il sistema prevede la presenza di un **terzo package** (**package *user***), che contiene le informazioni da scambiare tra il lato server e il lato client. Tale package dovrà contenere, quindi, tutte le informazioni che riguardano l'utente. In particolare, dovranno essere esposti il suo indirizzo SIP, in modo da poter instaurare con lui un canale di comunicazione, il suo AID (Agent IDentifier), per individuare la posizione dell'agente JADE in modo univoco sulla rete ed infine il suo stato.

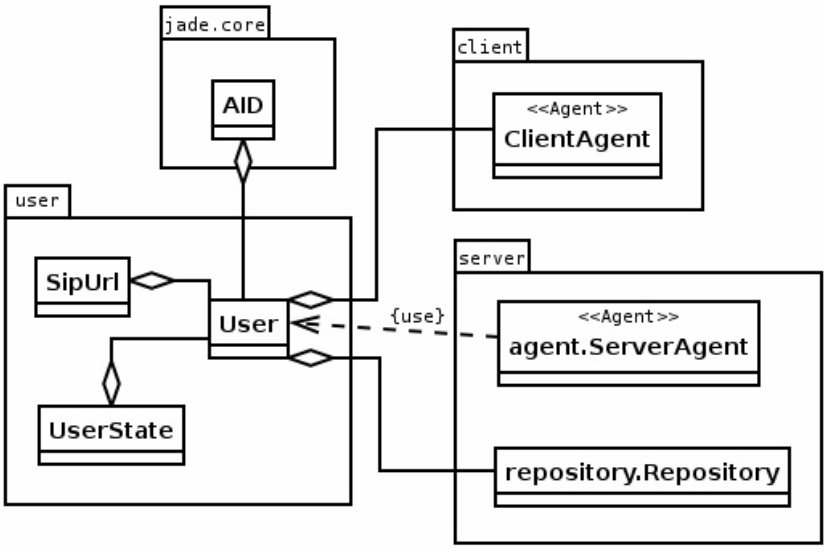

Figura 2.19 - Organizzazione globale del sistema

La figura 2.19 mostra il package *user* e le relative connessioni con i package discussi nei paragrafi precedenti.

Il package *user* si compone delle seguenti classi:

- *User*: rappresenta l'effettiva informazione di scambio, sia nelle interazioni client-server, sia nelle interazioni tra due Client Agent;
- *SipUrl*: contiene gli URL SIP degli utenti del sistema e serve per consentire la localizzazione di un utente sip nella rete;
- *UserState*: rappresenta lo stato dell'utente fra i cinque possibili: on-line, busy, away, invisibile, off-line. Gli stati possono essere usati per gestire le diverse politiche di comportamento degli agenti.

2.6.2 Funzionamento del Sistema

Data la natura del sistema JadeSip, che vede l'integrazione di due tecnologie eterogenee come il VoIP e gli agenti JADE, appare chiaro che il funzionamento del sistema si basi su due flussi distinti di dati: da un lato abbiamo i messaggi di controllo scambiati fra le diverse piattaforme JADE e, dall'altro lato, i messaggi SIP di instaurazione della chiamata e di trasporto dei media. La seconda categoria di messaggi è già stata descritta dettagliatamente in precedenza, quindi in questa sezione viene presentato il funzionamento del sistema dal punto di vista dei messaggi di controllo. Come scenario di esempio viene analizzata, anche con l'aiuto di un sequence diagram UML (v. figura 2.20), la registrazione di un agente al repository.

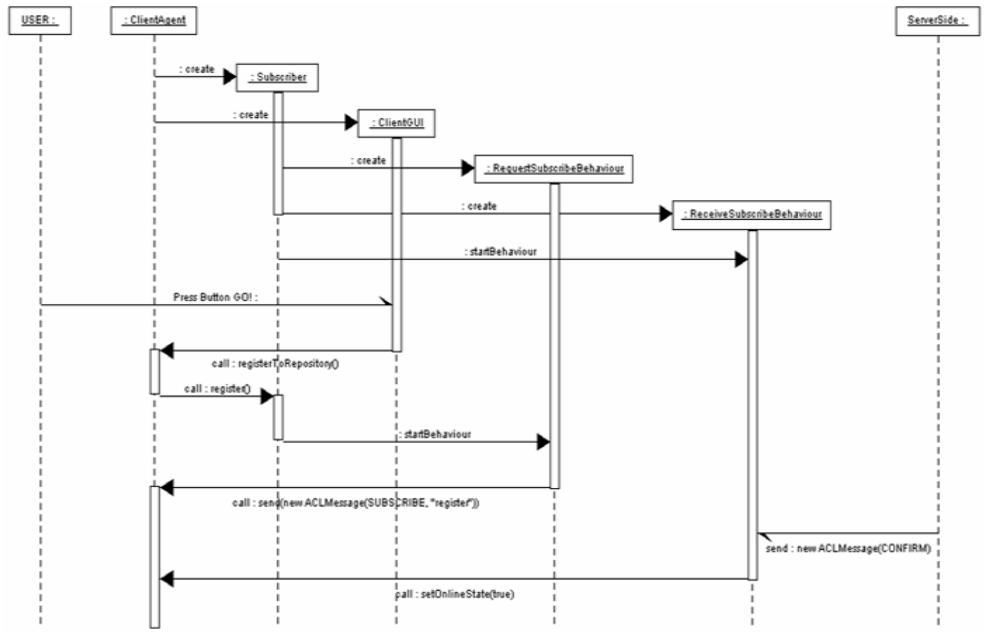

Figura 2.20 - Registrazione al repository

La registrazione avviene ad opera di due behaviour che cooperano tra di loro. Dapprima, il behaviour *RequestSubscribeBehaviour* invia la richiesta tramite il suo agente, mentre il secondo behaviour, *ReceiveSubscribeBehaviour*, riceve la risposta dal server e modifica lo stato dell'agente.

Come è possibile notare dall'esempio presentato, il sistema JadeSip fa un pesante utilizzo di messaggi ACL. L'interazione si basa su una serie di classi, molte delle quali sono behaviour JADE, ed ogni agente modifica il suo stato cooperando con l'ambiente esterno, sia esso un repository o un altro agente.

Quello che rimane da mostrare, anche nell'ottica di un ulteriore sviluppo del progetto, è l'insieme dei messaggi che vengono scambiati tra le entità del sistema, in modo da favorire una migliore comprensione di insieme dell'applicazione. In figura 2.21 sono mostrati i singoli behaviour dell'applicazione client e del sistema server, indicando per ognuno di essi quali sono i possibili messaggi ACL ricevuti ed inviati.

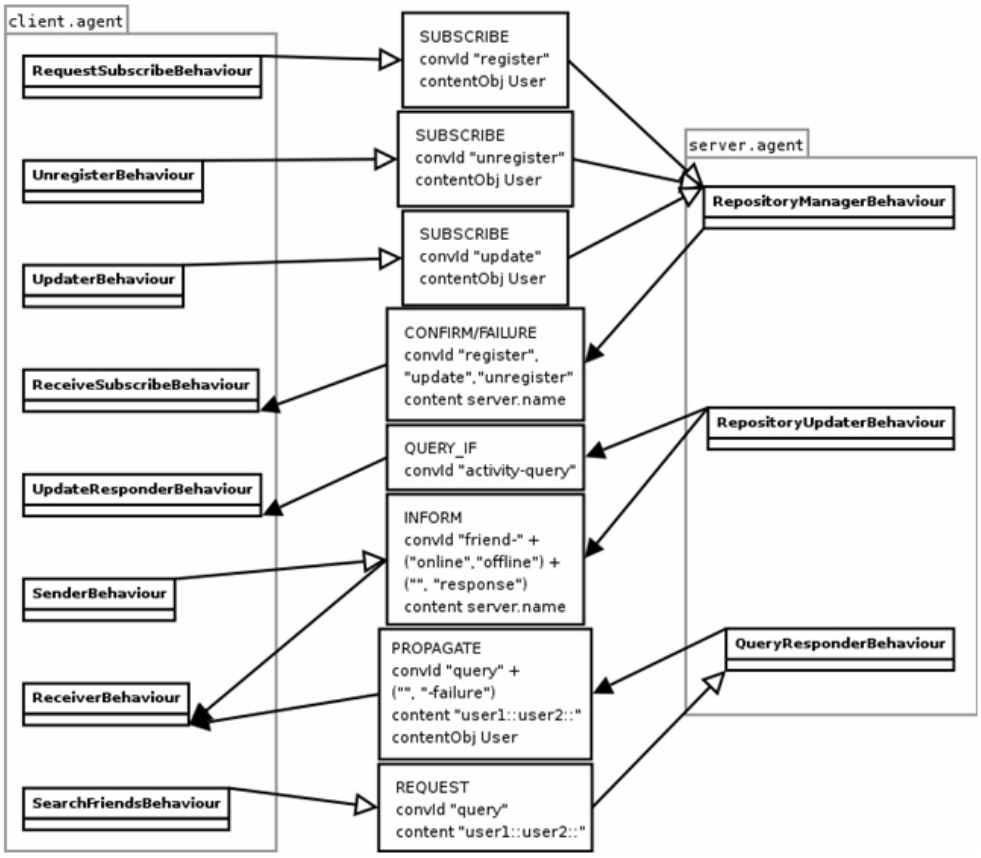

Figura 2.21 - Scambio di messaggi tra i behaviour

3
Tecnologia DTV-MHP

Questo capitolo presenta l'attività di ricerca incentrata sulla tecnologia DTV-MHP. In particolare, dopo una trattazione teorica degli standard, viene descritta l'interessante fase implementativa finalizzata allo sviluppo del sistema MHP-VC, basato sull'integrazione del concetto di "virtual community" all'interno della tecnologia DTV-MHP.

3.1 Introduzione

L'introduzione della televisione digitale (o DTV, Digital TeleVision) negli standard DVB (Digital Video Broadcasting), iniziata in Europa già nel 1994 con la diffusione diretta da satellite (DTH), sta cambiando profondamente gli scenari della comunicazione e del mercato. Delle tre tecnologie che si basano sui questi standard (satellite, cavo e diffusione terrestre), in Italia, data la scarsa diffusione delle prime due e la possibilità di passare alla trasmissione in digitale utilizzando le attuali infrastrutture analogiche, stiamo assistendo alla diffusione del DTT (Digital Terrestrial Television), che apre nuove possibilità sia per i produttori di servizi che per gli utenti. Lo standard di riferimento è il DVB-T (DVB-Terrestrial) e l'elemento di forte innovazione, dal punto di vista dell'interattività, è rappresentato da MHP (Multimedia Home Platform) [12], ambiente applicativo della TV digitale molto vicino alle applicazioni Internet e Web-oriented. Infatti il punto di forza del DTT, oltre ai vantaggi dati dall'aumento dei programmi trasmissibili per canale e

della qualità audio/video, è rappresentato proprio dell'interattività: viene introdotta la possibilità per l'utente di partecipare in modo attivo ed in tempo reale ai programmi televisivi (espressione di preferenze, selezione di prodotti, ecc.) con semplici azioni sul telecomando. Oltre all'interattività, con il DTT si ha la possibilità di usare il mezzo televisivo per la fruizione di servizi di informazione e di pubblica utilità generalmente accessibili solo tramite mezzi più complessi quali la rete Internet.

L'idea interessante che sta alla base di questa attività di ricerca è quella di sfruttare le potenzialità della tecnologia della televisione digitale integrando in essa il concetto di "virtual community", molto diffuso nella rete Internet: in questo modo si permette al concetto di interattività di evolversi passando dal paradigma utente-applicazione verso una più interessante e potente logica basata sull'interazione utente-utente o utente-comunità.

3.2 Lo Standard DVB-T

Attualmente il sistema televisivo tradizionale prevede l'utilizzo di segnali analogici: la trasmissione avviene in radio frequenza mediante onde elettromagnetiche, e la portante radio che definisce la banda di trasmissione (il canale) viene modulata in forma continua per trasmettere immagini in movimento e suoni. La televisione digitale terrestre, denominata tecnicamente DTT (Digital Terrestrial Television) o DVB-T (Digital Video Broadcasting - Terrestrial), utilizza invece segnali digitali: essi derivano dalla digitalizzazione (campionamento e quantizzazione) del segnale analogico. Ciò che ne deriva è una sequenza di bit che può essere ridotta grazie a tecniche di codifica e compressione, aumentando così la quantità di informazione che può essere trasmessa utilizzando le stesse risorse di banda.

Uno dei fattori che stanno determinando la diffusione della tecnologia DTT consiste nel fatto che, essendo il mezzo di trasmissione del tutto analogo a quello della televisione analogica (etere e radio frequenza), anche gli impianti di radioricezione della DTT sono identici a quelli utilizzati per la ricezione analogica.

Inoltre anche le bande di frequenza utilizzate sono identiche. Di conseguenza, le attuali antenne e la rete di distribuzione all'interno degli edifici, con gli opportuni dispositivi intermedi (derivatori, amplificatori, multiplexer e demultiplexer), sono già adatte alla ricezione digitale. In generale è anche previsto che le nuove reti digitali si avvalgano degli stessi siti di trasmissione della TV analogica; quindi, in teoria, non è necessario cambiare il puntamento della propria antenna. Ovviamente non deve essere installata alcuna parabola, necessaria invece per la TV satellitare: per la DVB-T è sufficiente l'antenna tradizionale con cui si ricevono le attuali emittenti analogiche nazionali e locali.

Oltre a questo vantaggio non indifferente, la tecnologia DVB-T offre una possibilità di sviluppo su due dimensioni:

- quantitativa, con l'aumento della capacità trasmissiva a parità di banda disponibile;
- qualitativa, caratterizzata dal miglioramento dei parametri di qualità del segnale e dall'ampliamento dei servizi connessi al servizio di broadcasting.

Il numero dei programmi che è possibile trasmettere in digitale è notevolmente maggiore: sulla stessa banda di frequenza in cui viene trasmesso un solo canale analogico, in digitale se ne possono trasportare oltre 5, grazie all'evoluzione delle tecniche di compressione utilizzate. Per quanto riguarda la qualità, la DTT consente la visione di programmi in formato 16:9, con audio multilingua di qualità CD, nonché l'utilizzo di nuovi servizi quali l'EPG (Electronic Program Guide).

Inoltre la tecnologia basata sullo standard digitale consente la trasmissione di canali dati che possono anche non essere correlati ai canali televisivi veri e propri con i quali condividono solo il mezzo trasmissivo. La disponibilità di un canale di trasmissione dati broadcast a velocità relativamente alta offre la possibilità di distribuire notevoli quantità di dati contemporaneamente a molti utenti. Questa funzione, non naturale per l'attuale architettura della rete Internet che è basata principalmente su connessioni punto-punto, apre nuove prospettive all'informatica. Unitamente alle grandi capacità di immagazzinamento di dati, oggi ottenibili a costi modesti, è facile pensare a intere basi dati domestiche aggiornate continuamente dal broadcast digitale e consultabili senza vincoli di tempo rispetto

alla loro diffusione. È evidente che solo la verifica a livello sperimentale potrà limitare la gamma di eventuali nuovi servizi che possono essere concepiti avendo la disponibilità di un tale tipo di risorsa.

DVB-T si inserisce in un contesto più ampio, il Digital Video Broadcasting (DVB) Project, un consorzio finalizzato allo studio e alla produzione di standard per la diffusione dei segnali televisivi.

3.3 DVB Project

Il Digital Video Broadcasting (DVB) Project è un consorzio di organizzazioni, sia pubbliche che private, operanti nel settore televisivo. Comprende oltre 270 imprese tra broadcaster, produttori hardware, operatori di rete, sviluppatori software ed enti regolatori sparsi in 35 stati diversi impegnati nella progettazione di standard globali per la trasmissione della TV digitale ed i suoi servizi correlati.

Gli standard del DVB, a livello di trasporto dei dati, si basano principalmente sul sistema di compressione MPEG-2 per la codifica audio e video. Il sistema di codifica audio si attiene allo standard MPEG layer II usato nel digital audio broadcasting. Lo standard MPEG-2 del sistema di codifica video accetta in ingresso quattro livelli da codificare: low level, main level, high 1440 level e high level, che si differenziano per la qualità offerta. Ognuno è caratterizzato da un certo bit-rate di sorgente. Come risultato della codifica, lo standard MPEG-2 offre differenti profili, ciascun profilo è caratterizzato da un set di strumenti di compressione. I profili sono cinque: simple profile, main profile, SNR scalable profile, spatially scalable profile e high profile. Ognuno di essi è progressivamente più sofisticato, e aggiunge strumenti di compressione al precedente. La codifica di sorgente MPEG-2 utilizzata da DVB è caratterizzata dall'uso della combinazione fra main profile e main level: *mainprofile@mainlevel*. Dopo la codifica di sorgente, i canali da trasmettere vengono accorpati in un unico stream mediante una operazione di multiplexing. Grazie a queste tecniche di compressione è possibile mettere molta più informazione sfruttando la stessa capacità di canale di quella analogica.

A livello fisico, DVB definisce tre standard: DVB-S, DVB-C ed il già citato DVB-T.

DVB-S è lo standard per la diffusione satellitare: è lo standard per le trasmissioni digitali satellitari. Per la ricezione richiede antenne paraboliche e decoder e la tecnica di modulazione utilizzata è la QPSK in banda Ku (11-14,5 GHz) e la copertura geografica delle trasmissioni è a livello continentale.

DVB-C è lo standard per la diffusione via cavo: è lo standard per trasmissioni digitali via cavo e utilizza canali di 8 Mhz all'interno della banda di downstream (tipicamente 70-130 MHz e 300-862 MHz). La modulazione utilizzata è la QAM con 16, 32, 64, 128 o 256 punti nel diagramma della costellazione. Per la ricezione lato utente è richiesto un decoder con eventualmente anteposto un cable modem per la separazione dei dati dal video e per la gestione del canale di ritorno. Tali livelli di modulazione permettono di ottenere bit-rate di 53 Mbps in un canale di 8 MHz.

DVB-T è lo standard per la diffusione terrestre: utilizza le frequenze nelle bande VHF/UHF permettendo di trasmettere fino a 4-5 canali digitali dove oggi può transitare un solo canale analogico (ampiezza di canale pari a 8 MHz). Non richiede nuove antenne ma gli utenti devono munirsi di nuovi decoder o nuovi televisori dotati di ricevitore digitale interattivo integrato. Consente di circoscrivere le trasmissioni a livello regionale e permette la ricezione anche a dispositivi mobili. La tecnica di modulazione utilizzata è quella numerica COFDM multiportante (Coded Ortogonal Frequency Division Multiplex), particolarmente robusta rispetto alle riflessioni del segnale. Combinando i vari schemi di modulazione e codifica si possono ottenere bit-rate che vanno da 4,98 Mbps a 31,67 Mbps.

Figura 3.1 - Loghi degli standard DVB-S, DVB-C e DVB-T

3.4 Decoder DVB-T

Per ricevere i segnali digitali forniti dalla trasmissione DVB-T è necessario possedere un decoder, detto anche set-top box (o STB), da collegare alla presa d'antenna ed al televisore tramite una presa SCART. Tramite il set-top box è possibile decodificare i segnali digitali ed utilizzare le applicazioni interattive associate ai programmi ed ai canali televisivi.

Figura 3.2 - Un classico decoder

I ricevitori DVB-T sono dispositivi molto complessi, composti da una circuiteria analogica dedicata alla demodulazione del segnale ricevuto, e da una parte digitale assimilabile ad un calcolatore elettronico. Dal segnale ricevuto e demodulato si ottiene uno stream binario che viene elaborato dalla circuiteria digitale. L'elaborazione del segnale digitale viene effettuata mediante un microprocessore ed il software ad esso associato.

Qualsiasi funzionalità implementata dal ricevitore è controllata via software, che è strutturato in due parti: una prima parte costituisce un vero e proprio sistema operativo che amministra le risorse hardware, mentre una seconda parte si occupa dell'implementazione delle varie funzionalità. Il software di gestione dei ricevitori può essere aggiornato da remoto permettendo il miglioramento delle funzioni già predisposte e la realizzazione di funzioni non previste all'atto dell'immissione sul mercato. La definizione di API mette a disposizione dei programmatori che devono aggiornare il software del ricevitore un'ampia gamma di librerie e routine che consentono l'implementazione di nuove applicazioni senza dover necessariamente conoscere in dettaglio e da un punto di vista troppo tecnico le specifiche dello standard DVB-T.

Esistono set-top box di due tipi:

- set-top box non interattivi (detti zapper), in grado di ricevere solo i programmi televisivi digitali;
- set-top box interattivi che, oltre a ricevere i programmi televisivi, sono in grado di usufruire dei nuovi servizi interattivi disponibili: questo modello viene spesso caratterizzato con la sigla MHP (v. sezione 3.6).

Nel caso dei decoder MHP, l'interattività viene garantita dal canale di ritorno: il decoder può collegarsi ad un provider di servizi in differenti modi, tramite modem V.90 o ISDN sulla linea telefonica tradizionale, oppure tramite modem a larga banda (ADSL) o ancora tramite rete GSM/GPRS.

3.5 Trasmissione Digitale e Transport Stream

In un sistema broadcast digitale, una tipica struttura trasmissiva potrebbe essere quella rappresentata in figura 3.3.

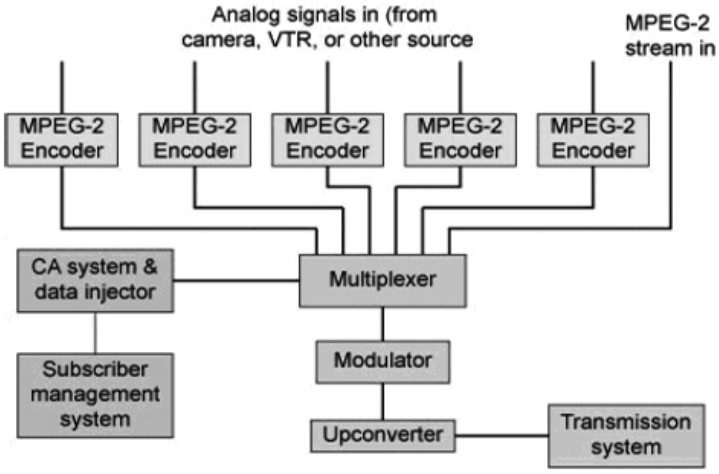

Figura 3.3 - La struttura trasmissiva presente nel sistema digitale terrestre

In tale struttura, gli encoder vengono utilizzati per convertire i segnali analogici in ingresso in formato digitale MPEG-2. Un encoder può generare due tipi di

stream MPEG. Il primo tipo è quello a bit-rate costante indipendentemente della complessità del segnale. Se il segnale è troppo complesso per essere codificato nel bit-rate specifico, la qualità viene ridotta; d'altra parte, se vengono utilizzati meno dati rispetto al bit-rate specificato, lo stream viene riempito con pacchetti nulli fino al raggiungimento del bit-rate corretto. Questa codifica rende l'elaborazione semplice, ma viene sprecata della banda. L'altro tipo di stream che può essere generato è quello a bit-rate variabile. In questo caso il bit-rate dello stream può essere modificato dinamicamente ed in questo modo la banda utilizzata varia in base alla complessità del segnale. Poiché alcune immagini hanno bisogno di maggiore banda per la codifica rispetto ad altre, la qualità viene mantenuta costante mentre l'occupazione di banda cambia. Lo stream a bit-rate variabile comporta una elaborazione un po' più complessa, ma fornisce maggiori vantaggi quando diversi stream MPEG vengono multiplexati insieme.

Il multiplexer prende uno o più stream MPEG e li associa in un singolo Transport Stream MPEG-2 (MPEG-2 TS). Ogni Transport Stream di solito ha a disposizione una larghezza di banda prefissata, che dipende dal mezzo di trasmissione e dalla rete trasmissiva. Il compito del multiplexer è inserire un insieme di servizi in questa banda. Il modo più facile per farlo è usare uno stream MPEG a bit-rate costante, poiché si sa esattamente quanta banda utilizza lo stream, quindi configurare un multiplexer risulta semplice.

I segnali digitali non possono essere trasmessi direttamente: prima devono essere modulati, quindi convertiti in segnali analogici per essere inviati utilizzando segnali radio o segnali elettrici su cavo. Le trasmissioni satellitare, terrestre e cablata possiedono caratteristiche differenti, perciò utilizzano modulazioni diverse.

Come detto in precedenza, il segnale televisivo digitale viene trasmesso come uno stream di dati MPEG-2, detto "Transport Stream" (TS). Il data rate di un TS è di circa 40 Mbps, sufficiente per sette/otto canali televisivi distinti. Un TS è composto da un insieme di "sotto-stream", detti "Elementary Stream" (ES), ognuno dei quali può contenere una traccia audio codificata MPEG-2, una traccia video MPEG-2, oppure alcuni dati incapsulati in uno stream MPEG-2. Ogni ES possiede

un PID (Packet IDentifier) che agisce da identificatore di pacchetto unico per quel dato ES all'interno del TS.

Per completezza di trattazione, di seguito viene descritto il processo di costruzione di un Transport Stream.

Si consideri un TS che contenga un singolo canale TV e si supponga di avere una sola traccia video ed una sola traccia audio. La prima operazione del processo di costruzione del TS consiste nella codifica del video e dell'audio nel formato MPEG-2 utilizzando un codificatore MPEG-2 (hardware oppure software). Alla fine della codifica si ottengono due ES, uno contenente la traccia audio codificata MPEG-2 ed uno contenente la traccia video. Questi due stream dovrebbero avere entrambi la stessa lunghezza, con le tracce contenute che iniziano dal medesimo istante, in modo che audio e video risultino sincronizzati. Successivamente sono codificati anche i dati in uno stream MPEG-2, utilizzando un tool che genera uno stream MPEG-2 a partire dalla directory che contiene i dati: questo processo produce un terzo ES. In questo caso non sussistono problemi temporali, poiché i dati vengono semplicemente ripetuti nello stream e non devono essere perfettamente sincronizzati con gli altri stream, a differenza di audio e video.

A questo punto sono state ottenute tutte le parti del canale TV, detto anche "servizio", codificate in ES MPEG-2 separati. Ora bisogna multiplexare i vari ES in un singolo TS utilizzando un multiplexer MPEG-2, che assegna PID ad ogni ES. Gli ES vengono suddivisi pacchetti, dove ogni pacchetto ha dimensione 188 byte e viene identificato tramite il PID. Il multiplexer prende questi pacchetti e li inserisce nel TS.

Ogni ES avrà un bit-rate differente: ad esempio sono necessari più dati per codificare un secondo di video piuttosto che un secondo di audio, quindi lo stream audio conterrà meno dati. Di conseguenza gli stream a bit-rate più alto, come quelli video, avranno più pacchetti inseriti dentro il TS finale per ogni pacchetto audio o dati che viene aggiunto.

Per far sì che il ricevitore sia in grado di ricostruire correttamente lo stream, devono essere aggiunte informazioni specifiche all'interno del TS: questi dati vengono codificati in alcuni ES che vengono aggiunti al TS durante la fase di

multiplexing, detti "Service Information" (SI). Il SI è un semplice database che descrive la struttura del TS tramite un certo numero di tabelle, ognuna delle quali descrive un servizio contenuto nel TS. Tali tabelle elencano gli ES che formano un servizio, evidenziano i PID degli ES ed i tipi di dati contenuti.

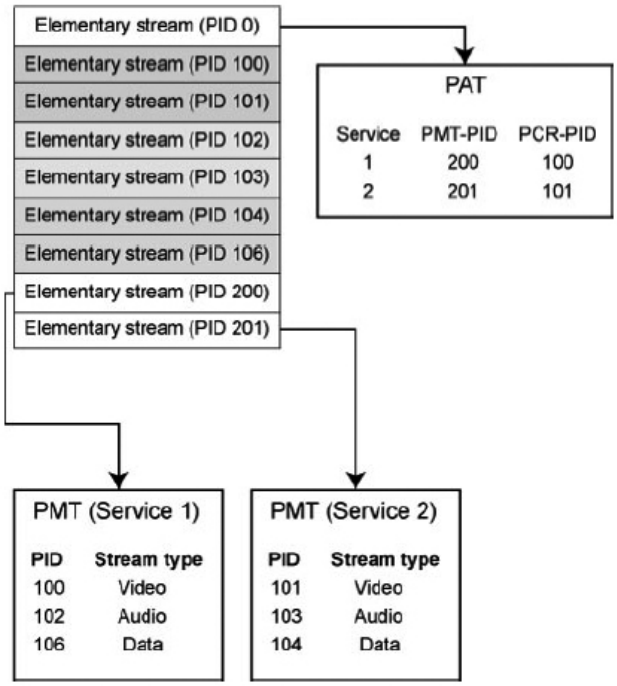

Figura 3.4 - Tipica struttura di un Transport Stream

Alcune tabelle SI descrivono i servizi specifici contenuti in un TS, mentre altre sono più generali e descrivono la struttura del TS stesso. Le tabelle SI più importanti che si trovano in un TS DVB sono il PAT (Program Association Table) ed il PMT (Program Map Table). Il PAT è l'unica tabella che viene trasmessa con PID fisso e mostra il PID dell'ES che contiene il PMT di un servizio. Il PMT è la tabella che descrive come è composto un servizio. Questa tabella descrive tutti gli ES di un servizio e notifica al ricevitore quale stream contiene il PCR (Program

Clock Reference) MPEG per il servizio. Il PMT non viene trasmesso con un PID prefissato ed esiste un PMT per ogni servizio contenuto nel TS.

Lo standard MHP, descritto nella seguente sezione, definisce una tabella SI extra chiamata AIT (Application Information Table). Questa tabella viene trasmessa come qualsiasi altro servizio e contiene la descrizione delle applicazioni MHP associate a quel servizio. L'AIT contiene tutte le informazioni di cui un ricevitore ha bisogno per eseguire le applicazioni e per dire all'utente quali applicazioni sono disponibili.

Le applicazioni MHP vengono trasmesse come parte di un TS MPEG-2 in un Object Carousel DSM-CC (DSM-CC è l'acronimo di "Digital Storage Media - Command and Control" e rappresenta lo standard per la trasmissione di dati basato sullo stream MPEG). L'AIT contiene tutte le informazioni necessarie per identificare l'applicazione all'interno dell'Object Carousel.

3.6 MHP - Multimedia Home Platform

Attualmente tutti i principali sevizi interattivi sono già stati implementati tramite tecnologie proprietarie (salvo alcune eccezioni) e sono attivi su diverse emittenti digitali, soprattutto Pay-TV satellitari. Il consorzio DVB ha standardizzato l'infrastruttura per il trasporto dati in ambiente televisivo, ma non ha inizialmente fornito alcuna specifica per la realizzazione di una qualunque piattaforma applicativa. Sono nate negli anni '90 alcune aziende che hanno sviluppato una propria tecnologia proprietaria per utilizzare i sistemi DVB per la distribuzione ed esecuzione di applicazioni sui ricevitori DVB; in questo panorama, un operatore di TV digitale intenzionato a fornire servizi interattivi si affida ad una tecnologia proprietaria per realizzare il proprio set di applicazioni supportate esclusivamente da un set-top box realizzato secondo quella tecnologia. Si parla di un mercato verticale, in cui sono presenti molteplici piattaforme per la fruizione dei servizi interattivi, quasi sempre incompatibili tra loro.

In questa moltitudine di soluzioni proprietarie si è avvertita la necessità di definire una piattaforma standard per la TV interattiva, soprattutto in vista della

fine delle trasmissioni televisive analogiche terrestri. Il consorzio DVB ha quindi provveduto alla creazione di un'architettura completa chiamata MHP (Multimedia Home Platform) [12] grazie alla definizione ed alla standardizzazione di un sistema di trasporto per le applicazioni, di un ambiente di esecuzione e di un set di API.

Da un mercato verticale si è quindi passati ad un mercato orizzontale: un'unica piattaforma standardizzata con cui poter accedere ai servizi offerti da molteplici broadcaster senza problemi di compatibilità. Le specifiche MHP coprono tutte le questioni relative alla realizzazione, alla trasmissione e all'utilizzo di applicazioni IDTV (Interactive Digital TV), nonché le caratteristiche peculiari del loro ambiente di esecuzione. MHP risponde alla necessità di avere una soluzione aperta che vada oltre a qualsiasi incompatibilità tra i sistemi proprietari attualmente già in uso.

Figura 3.5 - Logo dello standard MHP

MHP è uno standard relativamente giovane; la prima release è stata creata dal DVB Project e standardizzata dall'ETSI (European Telecommunications Standards Institute) nell'anno 2000. MHP definisce un interfaccia software generica (API) tra le applicazioni interattive provenienti dai vari fornitori di servizi e i terminali sui quali esse vengono eseguite, indipendentemente dalla loro implementazione hardware e software.

3.6.1 Architettura di Base

L'architettura MHP è definita da tre livelli:

- Resources: comprende tutte le parti essenziali del set-top box: decodifica MPEG, dispositivi di input/output, sottosistema grafico, ecc.;
- System Software: comprende un componente (Application Manager) per la gestione delle applicazioni ed il nucleo software (API) MHP, chiamato DVB-J (DVB-Java) e basato su una JVM (Java Virtual Machine);
- Applications: include le applicazioni interattive vere e proprie.

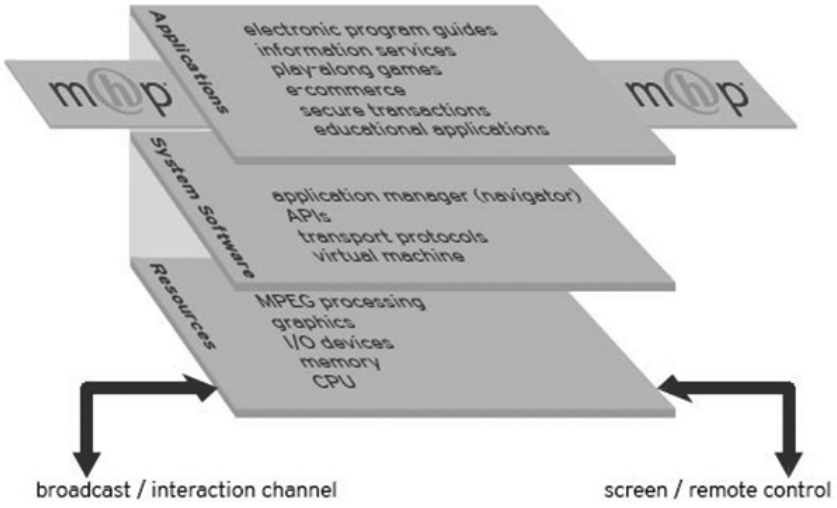

Figura 3.6 - Livelli MHP

3.6.2 Profili MHP

Il sistema MHP fornisce il concetto di profilo per aiutare lo sviluppo dello standard. Ogni profilo si riferisce ad una specifica area di applicazione definendo i requisiti che i set-top box devono soddisfare.

Attualmente esistono tre profili MHP, definiti in due set di specifiche:

- Enhanced Broadcast, definito nella specifica MHP 1.0 (ETSI ES 201 812);
- Interactive TV, definito anch'esso nella specifica MHP 1.0;
- Internet Access, definito nella specifica MHP 1.1 (ETSI TS 102 812).

Oltre alla definizione dei profili, gli standard MHP si occupano di altre questioni, come la piattaforma DVB-J (Java), i meccanismi MHP di sicurezza, i protocolli di download delle applicazioni, ecc.

Il profilo chiamato "Enhanced Broadcast" è il profilo base ed è stato progettato per mostrare le funzionalità del sistema middleware esistente e le possibili applicazioni da eseguire. Questo profilo non richiede ai set-top box capacità di gestione del canale di ritorno, ma consente solamente l'arricchimento dei contenuti audio/video con informazioni e immagini visualizzabili e navigabili sullo schermo.

Il profilo comprende la JVM, le API Java DVB, i protocolli di trasporto broadcast e le opzioni HTML.

Il profilo chiamato "Interactive TV" è il profilo intermedio che permette di utilizzare il canale di ritorno del set-top box per fornire servizi multimediali interattivi più complessi rispetto al profilo base. Questo profilo contiene API Java più appropriate per l'interattività ed il supporto al canale di ritorno, nonché protocolli di trasporto migliori.

Infine il profilo chiamato "Internet Access" richiede set-top box ancora più sofisticati, con un potere di elaborazione ed una memoria maggiori. Permette, tramite il canale di ritorno, di accedere a contenuti Internet. Questo profilo necessita di performance di alto livello essendo obbligatoria l'adozione di un browser Web e di un client e-mail embeddati nel set-top box. Contiene le API Java per l'accesso ad Internet, nonché un elemento opzionale chiamato DVB-HTML per la navigazione dei contenuti Web.

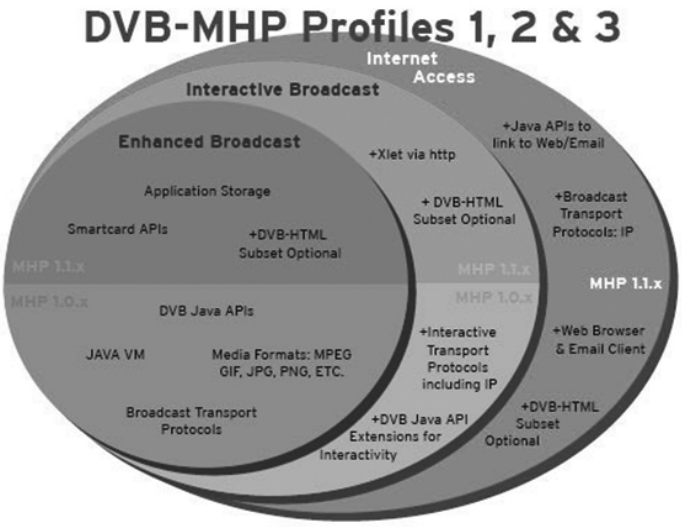

Figura 3.7 - Profili MHP

3.6.3 Stack MHP

Lo standard di base è costituito dalla specifica MHP 1.0 (ETSI ES 201 812), che comprende:

- l'architettura base di MHP;
- informazioni dettagliate sui profili Enhanced Broadcasting ed Interactive TV;
- diversi formati per i contenuti MHP, quali JPEG, MPEG-2 video e audio;
- protocolli di trasporto, che includono DSM-CC per la trasmissione broadcast e IP per il canale di ritorno;
- modelli di applicazione DVB-J;
- modelli di applicazione DVB-HTML;
- allegati al profilo DSM-CC, una presentazione testuale e varie API.

MHP 1.0.X specifica quindi l'ambiente dove si possono eseguire le applicazioni basilari per la TV digitale interattiva, indipendentemente dall'hardware e dal software specifici del produttore di set-top box.

In seguito è stata emanata la specifica MHP 1.1 (ETSI TS 102 812) per implementare il profilo Internet Access. Essa contiene:

- informazioni dettagliate sui profili Interactive TV ed Internet Access;
- la disponibilità per l'immagazzinamento delle applicazioni nella memoria persistente;
- download delle applicazioni mediante i canali broadcast o di interazione;
- estensioni DVB-J per un migliore supporto delle applicazioni;
- specifiche DVB-HTML;
- supporto per la gestione di plug-in interoperabili (per il supporto di formati di applicazioni non conformi);
- supporto per i riferimenti bidirezionali tra il contenuto di MHP ed il contenuto Internet.

La versione 1.1 di MHP è stata sviluppata sulla base delle specifiche e dei file sorgenti di MHP 1.0, con lo scopo di migliorare il supporto per l'utilizzo del canale di ritorno e consentire l'interoperabilità con i contenuti Internet.

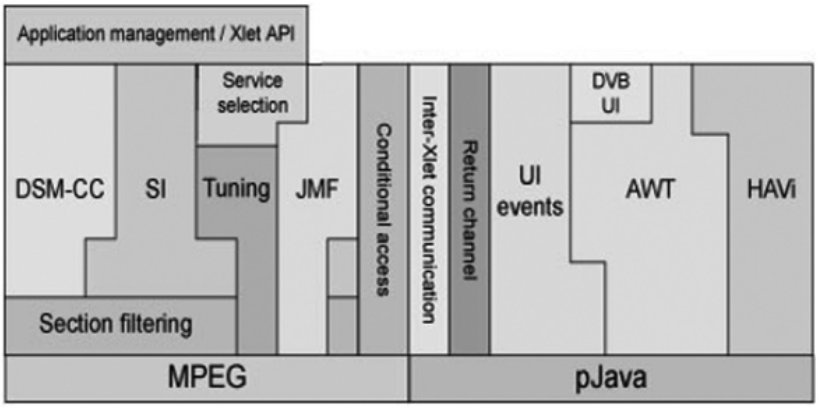

Figura 3.8 - Stack MHP

In figura 3.8 è mostrato lo stack software MHP, che, essendo progettato tramite una logica "a livelli" che vede la definizione di API MHP sopra altre API MHP, rende possibile un approccio modulare nella costruzione delle applicazioni.

Le API MHP possono essere suddivise in due categorie: API che si occupano dei servizi relativi agli stream MPEG ed API per lo sviluppo delle applicazioni.

Il secondo gruppo, appoggiandosi direttamente sulle API pJava (con pJava, acronimo di "Personal Java", si intende la versione di Java embeddata nei set-top box MHP), consentono l'implementazione dell'interfaccia grafica delle applicazioni MHP e la gestione dell'interazione con l'utente.

Le API principali che si occupano di MPEG sono le API Section Filtering: le altre API che riguardano MPEG sono costruite sopra di esse. Le API SI (Service Information) si occupano della gestione delle tabelle Service Information. Le API DSM-CC analizzano le sezioni DSM-CC ed utilizzano le API SI per cercare quali stream in un servizio contengono gli Object Carousel DSM-CC. Le API Service Selection utilizzano le API SI con lo scopo di individuare il servizio che deve essere sintonizzato; fatto questo, si utilizzano le API Tuning e JMF per sintonizzare il TS corretto e visualizzare il servizio. Infine le API Application Management, strettamente legate alle API Service Selection (ogni applicazione è associata ad un servizio) si occupano della gestione delle applicazioni MHP, di cui si parlerà nella sezione successiva.

3.6.4 Applicazioni MHP - Xlet

Come è possibile dedurre dall'analisi dello stack MHP descritto nella precedente sezione, il core software di MHP è basato interamente su Java: in particolare sono state adottate una Personal JVM (Java Virtual Machine) ed una serie di API, tra cui le API JavaTV di Sun Microsystems [30] ricoprono un ruolo fondamentale, in quanto forniscono la piattaforma di sviluppo per i servizi interattivi MHP.

Le API JavaTV sono state progettate per accedere alle funzionalità fornite dai ricevitori digitali, che comprendono lo streaming audio e video, l'accesso ai dati, i controlli per cambiare canale e quelli per l'interfacciamento con il televisore. Per la gestione dei contenuti multimediali vengono sfruttate le API JMF (Java Media Framework, di cui si è già parlato nella sezione 2.5.2), che definiscono varie funzionalità, quali la sincronizzazione dei media ed il controllo del ciclo di vita delle applicazioni.

L'ambiente hardware/software implementato all'interno di un ricevitore digitale include i seguenti livelli:

- Hardware Layer, definito dall'hardware del set-top box;
- Real Time Operative System (RTOS) Layer, livello che include il sistema operativo real-time ed i vari driver;
- Java Technology Layer, che include la piattaforma Java con le relative API JavaTV;
- Application Layer, dove vengono eseguite le applicazioni.

Le API JavaTV sono fondamentali in quanto forniscono un livello di astrazione che permette ai programmatori di non occuparsi dei dettagli specifici dell'hardware sottostante. Esse sono composte da numerosi package Java, il più importante dei quali è *javax.tv.xlet*, che contiene le classi che definiscono il ciclo di vita delle applicazioni. Alle API JavaTV mancano alcune funzionalità fondamentali fornite da altre API MHP, come il supporto per il canale di ritorno (fornito dalle API java.net), o il supporto per il controllo dei contenuti audio e video (fornito da JMF).

Le applicazioni sviluppate per MHP vengono chiamate applicazioni DVB-J o **Xlet**. Sono scritte in Java e consistono in un insieme di classi trasmesse in broadcast all'interno dello stream MPEG-2 ricevuto dal decoder.

Un'applicazione Java convenzionale per PC richiede un'esecuzione esclusiva tramite la JVM e classicamente si suppone che l'applicazione stessa abbia il controllo completo del proprio ciclo di vita. In un ricevitore televisivo digitale, invece, è frequente che più applicazioni vengano eseguite nello stesso momento, in questo senso possiamo assimilare il comportamento delle Xlet a quello delle applet Java, che possono essere avviate da una sorgente esterna, il browser, e non richiedono un'esecuzione esclusiva (è possibile che più applet siano avviate contemporaneamente all'interno di una medesima pagina Web). Ovviamente il sistema televisivo digitale è diverso dal Web, perciò devono essere adottati alcuni cambiamenti affinché questi concetti si adattino al ricevitore digitale.

Similmente alle applet, le applicazioni MHP permettono ad un software specifico esterno, chiamato "Application Manager" nel caso dei ricevitori digitali, di controllare il loro ciclo di vita, avviando o fermando la loro esecuzione. L'interfaccia *Xlet*, che si trova nel package *javax.tv.xlet*, contiene i metodi che permettono all'Application Manager di inizializzare, avviare, mettere in pausa e distruggere un'applicazione.

Insieme alla già citata *javax.tv.xlet.Xlet*, un'altra importante interfaccia definita dalle API JavaTV che consente di gestire il ciclo di vita e quindi l'interazione con l'Application Manager è chiamata *javax.tv.xlet.XletContext* e sarà descritta in seguito.

I quattro metodi definiti nell'interfaccia *javax.tv.xlet.Xlet* che consentono di gestire l'esecuzione di un'applicazione MHP sono i seguenti:

- *initXlet*: invocato dall'Application Manager per inizializzare la Xlet (tutte le inizializzazioni devono essere fatte nell'implementazione di questo metodo);
- *startXlet*: invocato per eseguire la Xlet;
- *pauseXlet*: invocato per mettere in pausa la Xlet;
- *destroyXlet*: invocato per terminare la Xlet.

Di conseguenza, il **ciclo di vita di una Xlet** è caratterizzato da 4 stati:

- Loaded: in questo stato la Xlet è stata creata ma non inizializzata, se durante questa fase viene rilevata un'eccezione allora si passa direttamente allo stato Destroyed. Una Xlet può trovarsi in questo stato solo una volta durante tutto il suo ciclo di vita;
- Paused: la Xlet è stata inizializzata e può trovarsi in questo stato dopo che il metodo *initXlet* è ritornato con successo dallo stato Loaded, oppure dopo che il metodo *pauseXlet* è ritornato con successo dallo stato Active. In questo stato la Xlet deve limitare al massimo l'utilizzo delle risorse condivise e soprattutto non impegnare la GUI televisiva;
- Active: in questo stato la Xlet è attiva e utilizza le risorse necessarie per fornire i suoi servizi. Se è dotata di GUI, allora deve essere l'unica applicazione abilitata a ricevere gli eventi dal telecomando;
- Destroyed: in questo stato la Xlet deve rilasciare tutte le risorse in uso per predisporsi alla terminazione.

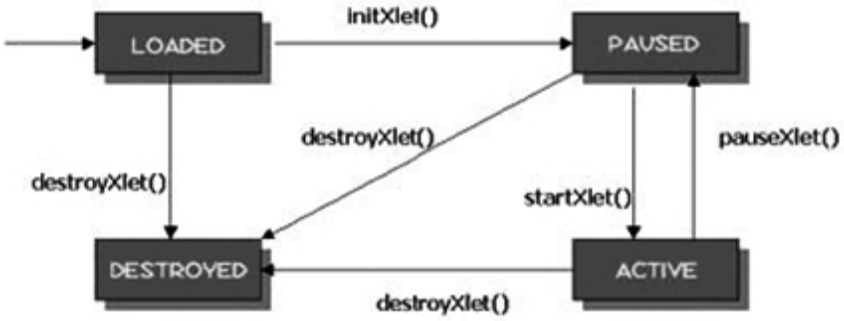

Figura 3.9 - Ciclo di vita di una Xlet

Una sequenza tipica di questo ciclo potrebbe essere:

1) l'Application Manager carica la classe principale della Xlet, su segnalazione del broadcaster, e ne crea un'istanza: l'applicazione si trova nello stato Loaded;
2) quando l'utente sceglie di avviare l'Xlet o un'altra applicazione segnala che la Xlet deve partire automaticamente, l'Application Manager invoca il

metodo *initXlet*. La Xlet viene inizializzata e carica le risorse (come le immagini). Quando l'inizializzazione è completa, la Xlet si trova nello stato Paused, ed è pronta per essere avviata;

3) una volta che il metodo *initXlet* ritorna con successo, l'Application Manager richiama il metodo *startXlet*. Questo comporta il passaggio dallo stato Paused allo stato Active, in cui la Xlet è in grado di interagire con l'utente ed utilizzare le risorse necessarie per fornire i propri servizi;
4) durante l'esecuzione, l'Application Manager può invocare il metodo *pauseXlet*: questo comporta il passaggio dell'applicazione dallo stato Active allo stato Paused. L'applicazione può tornare nello stato Active richiamando il metodo *startXlet*;
5) alla fine del ciclo di vita, l'Application Manager invoca il metodo *destroyXlet*, che comporta il passaggio allo stato Destroyed, liberando tutte le risorse. A seguito di questa operazione, l'istanza di questa Xlet non può essere più richiamata.

Ogni Xlet ha a disposizione un contesto di applicazione, detto *XletContext*, che viene usato per accedere alle proprietà dell'ambiente circostante e per comunicare i cambiamenti di stato all'Application Manager.

L'interfaccia *javax.tv.xlet.XletContext* contiene quattro metodi: *notifyDestroyed*, *notifyPaused*, *getXletProperty* e *resumeRequest*.

I metodi *notifyDestroyed* e *notifyPaused* permettono ad una Xlet di notificare al decoder che l'applicazione sta per terminare o per mettersi in pausa; in questo modo, il ricevitore conosce lo stato di ogni applicazione e può effettuare le azioni appropriate. Questi metodi devono essere richiamati immediatamente prima che l'Xlet entri negli stati Paused o Destroyed, in quanto il ricevitore potrebbe effettuare alcune operazioni per le quali l'applicazione non è detto che sia pronta.

Una Xlet può richiedere di passare dallo stato Paused allo stato Active usando il metodo *resumeRequest*. Questo può accadere quando si verifica un certo evento, rilevato ad esempio nello stream MPEG. Questo metodo richiede che un'applicazione venga fatta partire nuovamente, anche se il software del ricevitore potrebbe scegliere di ignorare questa richiesta a causa di limiti nelle risorse.

Il metodo *getXletProperty* permette alla Xlet di accedere alle proprietà segnalate dal broadcaster. Attualmente è stata definita una sola proprietà da JavaTV e da MHP, *XletContext.ARGS*, che permette ad un'applicazione di accedere agli argomenti che le vengono segnalati tramite AIT (Application Information Table).

Relativamente all'**interfaccia grafica**, ogni applicazione MHP, oltre ad implementare l'interfaccia *javax.tv.xlet.Xlet* di cui si è già parlato in precedenza, si basa su altri package, quali *java.awt*, *java.awt.event*, *org.havi.ui*, *org.havi.ui.event*, *org.dvb.ui* e *org.dvb.event*. Questi package servono per fornire le necessarie funzionalità grafiche ed i rispettivi eventi per implementare la GUI di interazione con l'utente.

Il modello grafico definito dallo standard MHP, come mostrato in figura 3.10, è basato su tre differenti livelli. Al livello più basso si trova il Background layer, che può contenere un colore o una immagine statica (rappresentata da un particolare frame MPEG-2). Il secondo livello è costituito dal Video layer, rappresentato dal flusso audio/video del canale TV o da una qualsiasi altra fonte in formato MPEG-2. Al livello più alto infine si trova il Graphic layer, che contiene l'interfaccia grafica della Xlet e può essere strutturato su più livelli sovrapposti.

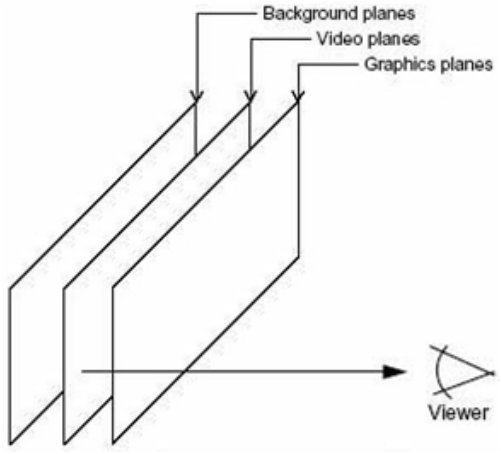

Figura 3.10 - Modello grafico definito da MHP

3.6.5 Fruizione di un Servizio Interattivo MHP

In figura 3.11 viene illustrata schematicamente la catena dell'interactive broadcasting sulla quale si basa la fruizione di un qualsiasi servizio MHP.

L'applicazione MHP (Java Xlet) ed i contenuti audio/video vengono multiplexati ottenendo un unico Transport Stream MPEG-2 (come discusso nella sezione 3.5), che viene broadcastato e successivamente ricevuto e decodificato dal set-top box, che si occupa di mandare in riproduzione i contenuti audio/video e contestualmente di avviare la Xlet.

In base all'interazione fra utente ed applicazione, viene attivato il canale di ritorno del set-top box per scambiare informazioni con un back-end server. In relazione alla tipologia di applicazione, queste informazioni possono consistere nella modifica del contenuto dell'applicazione stessa (es. servizio di televoto), oppure possono essere memorizzate all'interno di un database remoto per un utilizzo successivo (es. registrazione ad un portale).

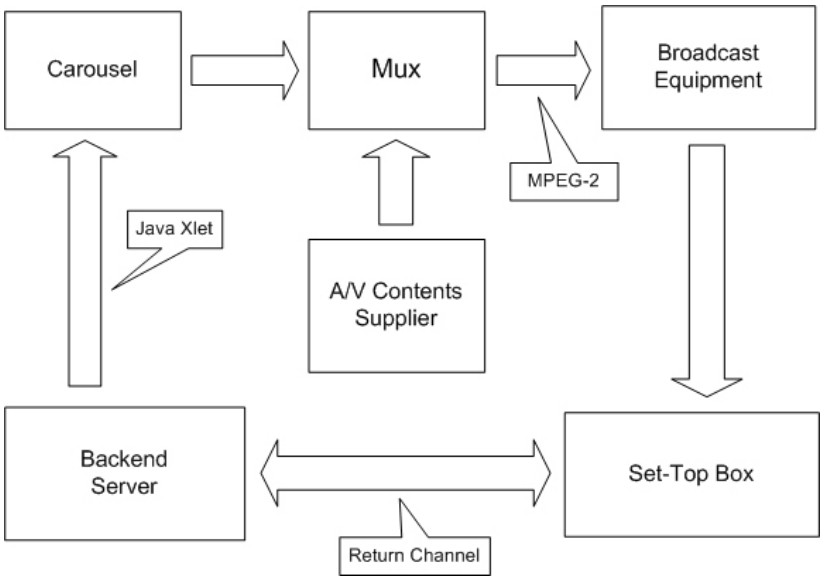

Figura 3.11 - Fruizione di un servizio interattivo per la DTV

Riguardo al trasporto dei dati, è importante ricordare che nella TV digitale interattiva l'MPEG-2 non rappresenta solo uno standard per l'encoding dei contenuti audio/video, ma è anche utilizzato come mezzo per il trasporto delle applicazioni. In particolare, come detto in precedenza, lo standard DVB ha ampliato lo standard MPEG-2 specificando come embeddare le applicazioni Java all'interno del Transport Stream MPEG-2.

Dal momento che le Xlet vengono broadcastate unitamente ai contenuti audio/video, per garantire all'utente la possibilità di cambiare canale ed accedere all'applicazione quando lo desidera (quindi ad un punto qualsiasi del programma visualizzato), è necessario che le Xlet vengano trasmesse in loop. Questa è la funzione dell'oggetto Carousel che compare nello schema.

3.6.6 Sviluppo di un Browser MHP

Per testare in termini pratici le funzioni di interattività che può fornire la televisione digitale terrestre, è stata inizialmente sviluppata un'applicazione MHP che implementa un semplice browser per la navigazione Web.

Il processo di sviluppo si è articolato in diverse fasi, dalla specifica dei requisiti con l'insieme dei vincoli hardware e software associati sia al sistema sia all'ambiente di sviluppo, ai processi di progettazione ed implementazione fino alle ultime fasi di revisione, validazione e verifica.

L'applicazione realizzata consiste in un browser Web per la navigazione di pagine HTML tramite tecnologia DTV-MHP. L'utente, tramite il proprio set-top box, è quindi in grado di visualizzare pagine Internet utilizzando un'interfaccia molto intuitiva utilizzabile tramite telecomando.

La fase di sviluppo è stata supportata da un'accurata fase di testing servendosi dell'emulatore XleTView [51], del quale in figura 3.12 viene mostrato il logo e l'interfaccia grafica. XleTView è un software open-source distribuito con licenza GNU GPL (General Public License).

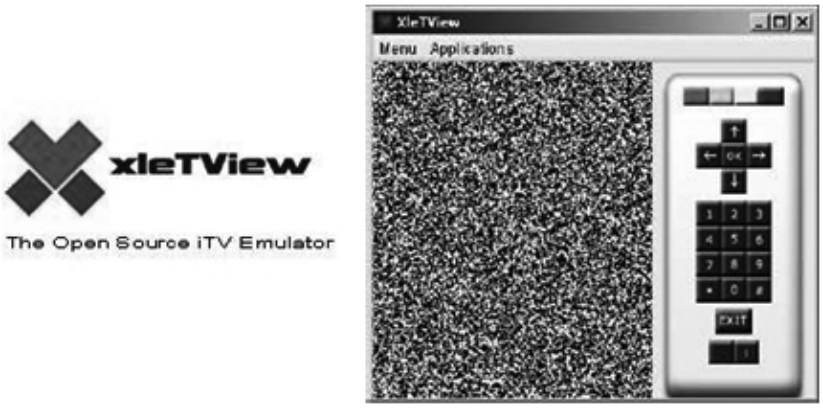

Figura 3.12 - Logo ed interfaccia di XleTView

Al termine delle fasi si sviluppo e di debugging, il software è stato testato direttamente su decoder mediante l'utilizzo di un particolare tipo di set-top box MHP-compliant in versione developer dotato di interfaccia e di software appositi per il testing in locale (ADB T75-DEV).

Figura 3.13 - Interfaccia grafica del browser MHP

L'applicazione sviluppata, della quale in figura 3.13 è mostrata l'interfaccia grafica, è in grado di interpretare i principali tag HTML consentendo all'utente di visualizzare immagini e tabelle, compilare form ed accedere a link, garantendo una navigazione base ma agevole delle pagine Web.

3.7 Framework MHP-VC

Nonostante il grande interesse di ricerca suscitato negli ultimi anni e l'elevato numero di funzionalità e di servizi ormai offerti, attualmente i gruppi di ricerca che lavorano sullo standard MHP stanno concentrando i propri sforzi soprattutto sulla personalizzazione dei contenuti sulla base dell'analisi dei profili e delle preferenze degli utenti. Ciò che emerge è una totale assenza di aspetti collaborativi, ossia non è possibile ritrovare al momento progetti finalizzati all'integrazione, all'interno della tecnologia DTV, di servizi al supporto di gruppi di utenti accomunati da particolari tipi di interessi o necessità. Questa tipologia di servizi è molto comune sulla rete Internet, basti pensare al pressoché infinito numero di forum, di blog o di servizi generali che permettono un'interazione diretta tra due o più utenti (es. giochi multi-utente).

Sulla base di queste considerazioni, ha preso il via un progetto di ricerca con l'obiettivo di arricchire la tecnologia DTV basata sullo standard MHP descritta nelle sezioni precedenti con il concetto di "virtual community" molto comune sulla rete Internet. Il risultato è stato lo sviluppo del framework MHP-VC [16][17] (dove VC è l'acronimo di virtual community) che, come vedremo, consente di sviluppare servizi MHP basati sull'interazione diretta di due o più utenti, modificando in questo modo il concetto di interattività MHP da un piano utente-applicazione verso una più interessante logica utente-utente o utente-community.

3.7.1 Definizione di Virtual Community

L'enorme diffusione che ha avuto la rete Internet ha portato non solo ad un aumento esponenziale delle possibilità di comunicazione, ma ad una vera e propria

rivoluzione delle abitudini e delle relazioni fra le persone. Internet, come oggi la conosciamo, non è più un semplice mezzo utile per velocizzare e migliorare le comunicazioni: lo è stato all'inizio della sua ascesa, con l'accelerazione della velocità di scambio di messaggi portata dalla e-mail, o con la trasformazione del modo di pubblicizzare, far conoscere, ricercare informazioni introdotta con il Word Wide Web. Se guardiamo Internet oggi, non vediamo solamente uno strumento o un insieme di mezzi, ma più uno spazio virtuale, nel quale nascono e si evolvono relazioni e rapporti interpersonali, nel quale è presente una sorta di società parallela.

Ma cosa si intende con il termine virtual community? Una virtual community rappresenta un concetto multidisciplinare, difficile da definire, che risulta da molte definizioni, ognuna dipendente dalla prospettiva dalla quale la si guarda.

H. Rheingold, nel proprio libro [20] scritto ispirandosi largamente alla propria esperienza diretta, reale e virtuale, di relazioni personali all'interno della prima e da allora più famosa tra le comunità on line, The Well, definisce le comunità virtuali come "aggregazioni sociali che emergono dalla rete quando un certo numero di persone porta avanti delle discussioni pubbliche sufficientemente a lungo, con un certo livello di emozioni umane, tanto da formare dei reticoli di relazioni sociali nel cyberspazio". A sua volta C. Figallo, per anni gestore di The Well, propone una serie di requisiti più stringenti e più formalizzati: i membri si sentono parte di un insieme sociale unitario, c'è una ragnatela di relazioni tra i membri, c'è uno scambio continuo di informazioni cui si attribuisce un valore comune, le relazioni tra i membri durano nel tempo creando delle storie condivise. In altre parole c'è un'identità collettiva in cui riconoscersi. Ma questo non è un punto di partenza dato fin dagli inizi. In qualche caso i fini sociali saranno stati definiti con chiarezza dai fondatori (ad esempio definendo il topic di quella comunità: "ci occupiamo di informatica"), in altri, e sono i più frequenti, sarà il gruppo che si riunisce spontaneamente attorno ad uno stimolo iniziale a far emergere punti di vista comuni, valori, ed obiettivi. Nei casi migliori questo finirà per generare anche linguaggi di gruppo e riti sociali, anch'essi patrimonio di quella particolare comunità e della sua storia.

C. Romm ed R. J. Clarke [6], ponendo grande attenzione sul mezzo comunicativo, definiscono le virtual community come gruppi di persone che comunicano fra di loro tramite mezzi elettronici, come Internet, per condividere interessi comuni non vincolati dalla posizione geografica, dalle interazioni fisiche o dalle origini etniche. J. Hagel ed A. Armstrong [25] danno una prospettiva di business e definiscono le virtual community come gruppi di persone tenute insieme da un'opportunità di condividere un senso di comunità con estranei con interessi comuni. Una definizione piuttosto classica è quella fornita da Q. Jones e S. Rafaeli [43], nella quale le virtual community sono definite "virtual public", spazi virtuali aperti che permettono a gruppi di persone di collaborare e di contribuire alla crescita di interazioni e legami personali.

In letteratura è possibile reperire molte altre definizioni, ma è possibile individuare alcuni elementi comuni:

- le virtual community sono comunità virtuali i cui membri, a differenza delle comunità tradizionali, sono fisicamente lontani ed entrano in relazione solo tramite una sorta di cyberspazio comune;
- le virtual community sono basate su una tecnologia che rende possibile l'interazione fra i propri membri: le diverse definizioni fanno sempre riferimento più o meno diretto al fatto che l'accesso ad una virtual community sia sempre possibile tramite un computer o uno strumento simile;
- le virtual community si formano attorno ad uno scopo centrale ed a un interesse condiviso, che può spaziare dal semplice divertimento alla condivisione di informazioni e/o conoscenze, alla creazione di relazioni sociali;
- il successo di una virtual community è strettamente legato all'attività dei propri utenti, per questo motivo la crescita di una virtual community è solitamente graduale.

In conclusione possiamo definire una virtual community come uno spazio virtuale basato su una specifica tecnologia che consente la comunicazione e

l'interazione fra gli utenti, accomunati generalmente da un particolare interesse o obiettivo.

3.7.2 Virtual Community e MHP

La televisione digitale ha portato un'enorme innovazione, anche concettuale, nella visione classica della TV. Il telespettatore infatti passa da una posizione totalmente passiva, in cui è solo consumatore di contenuti preparati ed erogati dalle emittenti televisive, a fruitore attivo non solo di contenuti selezionabili e personalizzabili, ma anche di servizi interattivi precedentemente utilizzabili solo con altri mezzi tecnologici.

Quello che al momento è completamente assente nell'infrastruttura DTV-MHP è l'aspetto collaborativo: l'interattività in MHP è semplicemente intesa come interazione fra utente ed applicazione.

Da questa considerazione ha preso il via un'attività di ricerca con l'obiettivo di ampliare gli orizzonti della tecnologia MHP tramite l'inserimento di una nuova gamma di servizi al fine di consentire un'interazione diretta fra i vari utenti. L'obiettivo perseguito e raggiunto è stato quello di integrare nella tecnologia DTV-MHP il concetto di virtual community tramite lo sviluppo un framework software per l'implementazione di applicazioni MHP al supporto di comunità di utenti impegnati in attività collaborative.

Rifacendosi alla definizione di virtual community fornita al termine della sezione precedente, possiamo affermare che l'attività di ricerca ha focalizzato il proprio interesse soprattutto sull'elemento tecnologico, dotando il concetto di virtual community, prima legato quasi essenzialmente al mondo del Web, di una nuova tecnologia di comunicazione molto più user-friendly come la TV digitale interattiva basata su MHP.

La possibilità di integrare la tecnologia DTV-MHP con l'idea di virtual community può fornire due grandi vantaggi: da un lato abbiamo l'incremento delle potenzialità della TV digitale, con la creazione di una nuova gamma di servizi innovativi, dall'altro lato, utenti che non possiedono conoscenze informatiche sufficienti per navigare sul Web hanno la possibilità di beneficiare di servizi tipici

delle virtual community tramite DTV. Possiamo quindi affermare che l'integrazione della televisione digitale con il paradigma delle virtual community estenda il concetto base di interattività MHP, movendolo da una semplice logica utente-TV a una più interessante logica basata sull'interazione utente-utente e utente-comunità.

Basandoci sulle definizioni e sulle caratteristiche delle virtual community analizzate nella sezione precedente, possiamo da subito definire una serie di funzionalità di base che il framework deve offrire per supportare una comunità:

- profilazione utente: ad ogni utente deve essere associato un profilo personale nel quale sono memorizzate informazioni utili per ottimizzare l'iterazione con gli altri componenti della comunità;
- accesso controllato: l'accesso alla piattaforma a supporto della comunità deve essere permesso solo agli utenti registrati;
- comunicazione utente-utente e utente-comunità: deve essere possibile un qualche tipo di comunicazione sia fra i vari utenti, sia fra un utente e l'intera comunità;
- aggiornamento dello stato: i contenuti "pilotati" dagli appartenenti alla comunità devono essere disponibili in versione aggiornata a seconda del tipo di servizio che si vuole creare (ad esempio per un community game l'aggiornamento dello stato deve avvenire in tempo reale).

3.7.3 Il Framework

Lo sviluppo del framework si è basato sulla tecnologia ad agenti, che, per le caratteristiche intrinseche degli agenti stessi (come la proattività), si adattava molto bene allo scopo del progetto.

Come discusso nel primo capitolo della presente tesi, gli agenti necessitano di particolari risorse per comunicare. Tali risorse, secondo le specifiche FIPA [18], consistono in una piattaforma che fornisce loro i servizi base per interagire l'un l'altro (comunicazione tramite messaggi, servizio di reperimento degli agenti, ecc.). La comunicazione può avvenire fra agenti appartenenti alla medesima piattaforma ma anche tra agenti dislocati in piattaforme diverse, purché i messaggi

siano conformi alle specifiche FIPA. Attualmente sono disponibili piattaforme FIPA per diverse tipologie di dispositivi [5][14][15][26], il framework sviluppato si propone di estendere le potenzialità della tecnologia ad agenti permettendone lo sviluppo anche su dispositivi DTV basati su Java come i set-top box MHP.

Il framework consiste in un sistema ad agenti composto da una parte server e da una parte client. La parte server, che consiste in una piattaforma ad agenti standard, risiede su un Web server ed è stata sviluppata secondo le specifiche classiche FIPA. La parte client rappresenta invece la componente più innovativa, in quanto risiede sul set-top box MHP e si propone di abilitare la tecnologia ad agenti su questa tipologia di dispositivi.

Nelle sezioni seguenti vengono descritte nel dettaglio le due componenti del framework e vengono presentati alcuni esempi di servizi implementati.

3.7.4 Lato Client

La componente client, come detto in precedenza, risiede sul set-top box dell'utente. La scelta è stata quella di integrare la parte client all'interno di una classica applicazione MHP. In figura 3.14 è schematizzata l'architettura del client, nel quale sul classico stack MHP si appoggia la piattaforma ad agenti che consente di sviluppare le varie applicazioni.

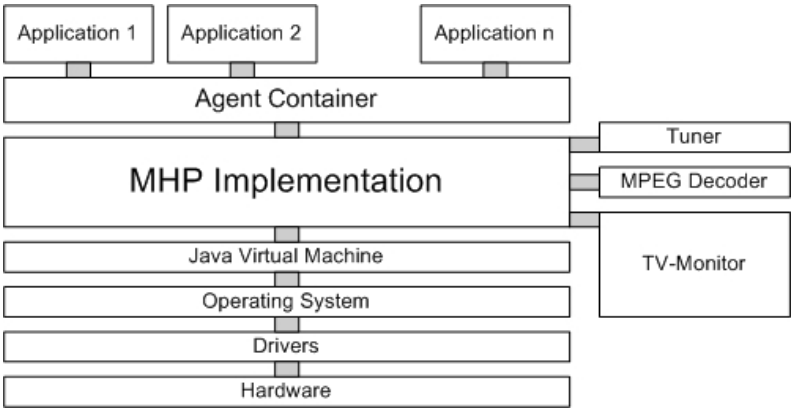

Figura 3.14 - Architettura lato client

Nello standard MHP, le applicazioni vengono eseguite in relazione all'utilizzo di particolari servizi interattivi da parte dell'utente e, solitamente, vengono terminate alla chiusura del servizio. Nell'ottica di sviluppare servizi per virtual community, è necessario che il sistema sia in grado di rilevare e memorizzare le caratteristiche e le preferenze dell'utente relative al particolare servizio (es. profilo utente all'interno di un community game). In questo senso in fase progettuale si è pensato di integrare all'interno del client un particolare agente, chiamato User Agent, che opera come interfaccia fra l'utente ed il resto del sistema e si occupa di raccogliere e di memorizzare le caratteristiche dell'utente stesso.

Lo User Agent è quindi responsabile della creazione e dell'aggiornamento del profilo dell'utente e comunica con l'utente stesso tramite una semplice GUI su TV. La comunicazione fra lo User Agent e gli altri agenti che risiedono sulla componente server del framework avviene tramite messaggi FIPA standard scambiati sul canale di ritorno del set-top box MHP.

Per consentire lo sviluppo di servizi per virtual community, il framework è stato inizialmente progettato basandosi sul fatto che lo User Agent fosse sempre attivo sul dispositivo dell'utente. Questo avrebbe consentito all'utente di delegare particolari task all'agente anche nel momento in cui non si fosse trovato in condizioni di utilizzare attivamente il set-top box. In realtà, come vedremo successivamente, per mantenere la componente client più leggera possibile (una sorta di "thin client"), si è pensato, almeno nella prima release del sistema, di delegare tutta la business logic dei servizi alla parte server. Questo significa che il lato client del framework rappresenta semplicemente un'interfaccia fra l'utente e la parte server alla quale sono delegati tutti i meccanismi di ragionamento e di gestione dei servizi.

3.7.5 Lato Server

La componente server del framework, come affermato in precedenza, consiste in una piattaforma ad agenti JADE collocata su un Web server standard. La comunicazione fra le due componenti del framework è garantita dal canale di ritorno del set-top box MHP.

Il sistema include quattro differenti tipi di agenti: SP Agent (Set-top box Proxy Agent), MP Agent (Mux Proxy Agent), Service Agent, User Profile Manager e Directory Facilitator.

L'agente chiamato "SP Agent" rappresenta l'interfaccia fra il sistema multi-agente lato server e la componente client del framework: il suo compito è quello di ricevere i messaggi FIPA provenienti dal client, processarli, inoltrarli agli altri agenti lato server e rispondere al client affinché lo stato del servizio possa essere aggiornato.

L'agente chiamato "MP Agent" è l'unico agente previsto nella progettazione iniziale del framework ma non effettivamente implementato. Questo agente avrebbe dovuto gestire l'interazione tra il Web server e la parte di broadcasting dei contenuti per l'aggiornamento dello stato dell'applicazione all'interno del flusso MPEG-2 inviato al decoder. Tuttavia, nell'implementazione della prima release del sistema, l'aggiornamento viene gestito totalmente tramite il canale di ritorno che collega il decoder al Web server.

L'agente chiamato "Service Agent" ha un ruolo fondamentale all'interno del framework, in quanto si occupa della gestione di uno specifico servizio offerto agli utenti. Questo significa che, per ogni applicazione sviluppata, si avrà un particolare Service Agent creato appositamente per la sua gestione. Ad esempio, relativamente ad un community game avremmo un Service Agent preposto all'aggiornamento dello stato del gioco, alla ricerca di un avversario adatto all'utente, ecc.

Infine abbiamo gli agenti chiamati "User Profile Manager" e "Directory Facilitator". Il primo si occupa della gestione dei profili degli utenti in relazione agli specifici servizi offerti, mentre il secondo è un agente standard della piattaforma JADE che fornisce un servizio di pagine gialle.

La figura 3.15 mostra una rappresentazione grafica dell'architettura globale del sistema sottolineando le interazioni fra le diverse tipologie di agenti. Come sempre, la sovrapposizione di più agenti suggerisce che nel sistema può essere presente contemporaneamente più di un agente di quel tipo.

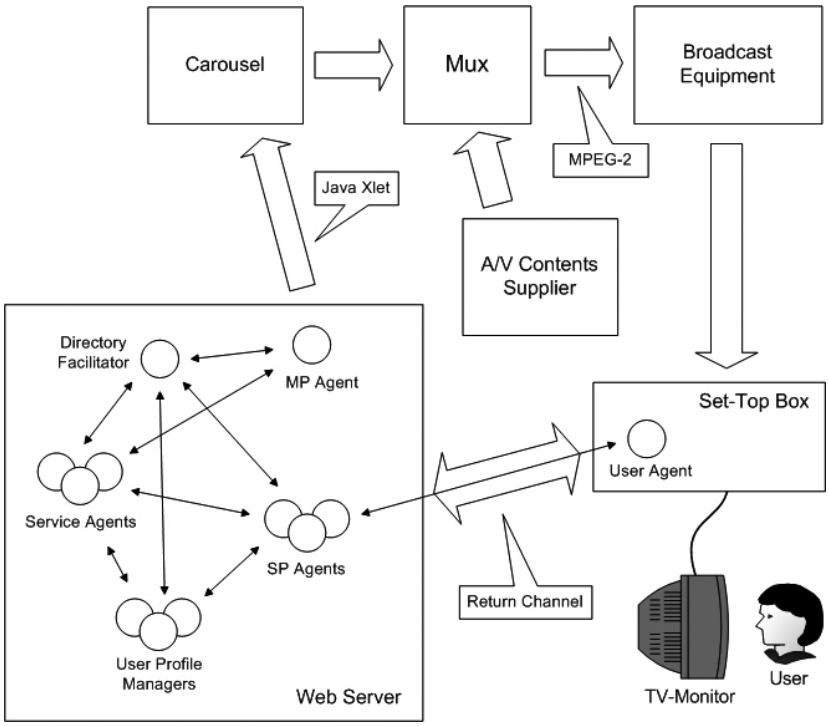

Figura 3.15 - Architettura globale del framework MHP-VC

L'architettura rappresentata in figura 3.15 è relativa alla progettazione iniziale del framework. Tuttavia, come detto in precedenza, nella prima implementazione del sistema è stata tralasciata la parte di comunicazione fra Web server e sistema di broadcasting e la gestione dello stato dei servizi viene completamente effettuata tramite il canale di ritorno del set-top box.

3.7.6 Implementazione di un Community Game

L'idea di giocare on-line con altri utenti connessi ad una rete, o, in generale, che utilizzano una tecnologia che supporta interazioni real-time fra i partecipanti al gioco, è molto comune e largamente diffusa su Internet. Il primo servizio sviluppato sulla base del framework MHP-VC si propone di integrare questa idea con la tecnologia della televisione digitale, permettendo agli utenti MHP di prendere parte ad un community game.

Si è scelto di utilizzare un tipo di gioco relativamente semplice, come “Othello”, che richiede due giocatori, integrato in un’architettura che permetta l’incontro fra diversi utenti (gli appartenenti alla comunità). L’utente MHP, utilizzando il telecomando del decoder interattivo, ha la possibilità di accedere, utilizzando le proprie credenziali, allo “spazio” della comunità. Si ritrova quindi in un ambiente virtuale nel quale incontrare gli altri partecipanti alla community, ossia gli altri giocatori: da qui può scegliere se giocare una partita o guardarne una già in corso.

Il componente lato client del community game è la Xlet MHP che viene caricata sul set-top box e si occupa dell’interazione con l’utente (gestione degli eventi del telecomando) e della comunicazione con il server (gestione dello scambio di messaggi con il back-end server). Per sviluppare e testare l’applicazione è stato utilizzato l’emulatore XleTView [51], di cui si è già parlato in precedenza.

Il modello seguito nello sviluppo della Xlet è quello utilizzato in molti community game presenti su Internet, e consiste in una “virtual place” nella quale avviene l’incontro fra i giocatori, la scelta dell’avversario e/o la creazione di nuovi match di gioco. Ogni partita creata dai vari utenti è chiamata “stanza”, poiché con essa si vuole rappresentare, più che il semplice incontro di gioco, un ambiente nel quale sono presenti più giocatori, alcuni che giocano attivamente, altri che fanno da spettatori.

Lo scenario considerato si presenta nel seguente modo:

- l’utente effettua il log-in utilizzando le proprie credenziali (username e password);
- se il log-in ha successo, l’utente viene indirizzato in una schermata che contiene l’elenco degli utenti connessi e delle partite già create: da qui è possibile scegliere se entrare in una stanza in cui è presente un solo giocatore e iniziare il gioco, decidere di assistere ad una partita già in corso, o creare una nuova stanza;
- se si decide di entrare in una stanza o di crearne una nuova, viene presentata una schermata in cui sarà possibile iniziare il gioco (se sono presenti due giocatori), attendere un avversario o invitare un giocatore ad unirsi alla stanza.

La Xlet È facilmente intuibile che nel primo servizio sviluppato la parte client non presenta funzionalità tipiche della tecnologia ad agenti. Per velocizzare l'implementazione del servizio si è pensato di delegare al server la completa logica del gioco e, dato che un gioco di community non richiede nemmeno particolari meccanismi di delega, l'unica caratteristica "ad agenti" del client è rappresentata dal linguaggio che utilizza per comunicare con il Web server, basato su messaggi ACL FIPA.

Per mostrare il comportamento del sistema relativamente allo svolgimento di una partita, nelle seguenti figure vengono presentate alcune schermate dell'interfaccia grafica con cui l'utente MHP si trova ad interagire.

Nel momento in cui l'applicazione client viene eseguita dal set-top box, sullo schermo TV appare la prima schermata che permette l'autenticazione dell'utente per mezzo di username e password (figura 3.16).

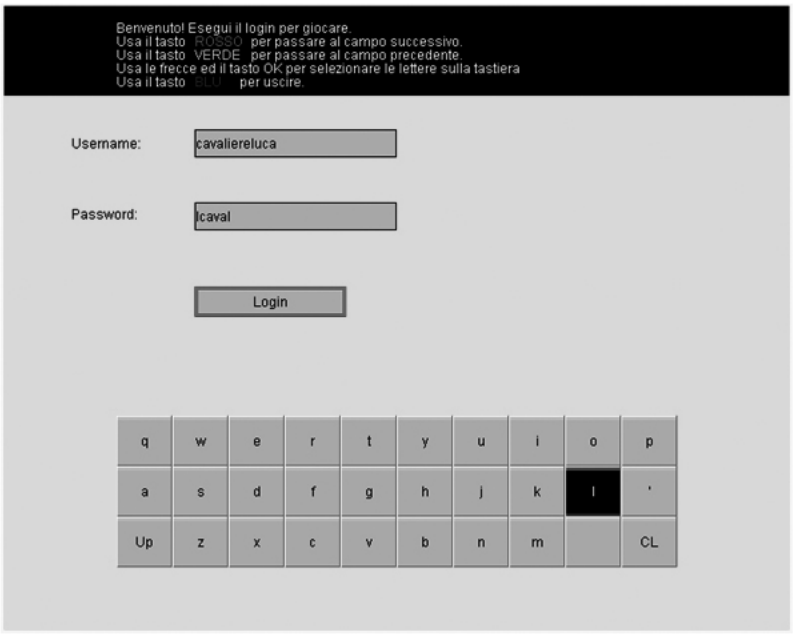

Figura 3.16 - Schermata iniziale di log-in

Tramite il telecomando e la tastiera su schermo l'utente inserisce le credenziali e preme il tasto *Login*. Nell'architettura globale del sistema questo significa che, lato server, il Service Agent preposto alla gestione del servizio Othello riceve una stringa con i dati forniti dall'utente e, se sono corretti, notifica al client che l'utente è abilitato all'accesso.

Ovviamente il Service Agent è anche preposto alla gestione dei dati visualizzati nelle schermate successive nonché all'aggiornamento dello stato del gioco mano a mano che la partita viene disputata.

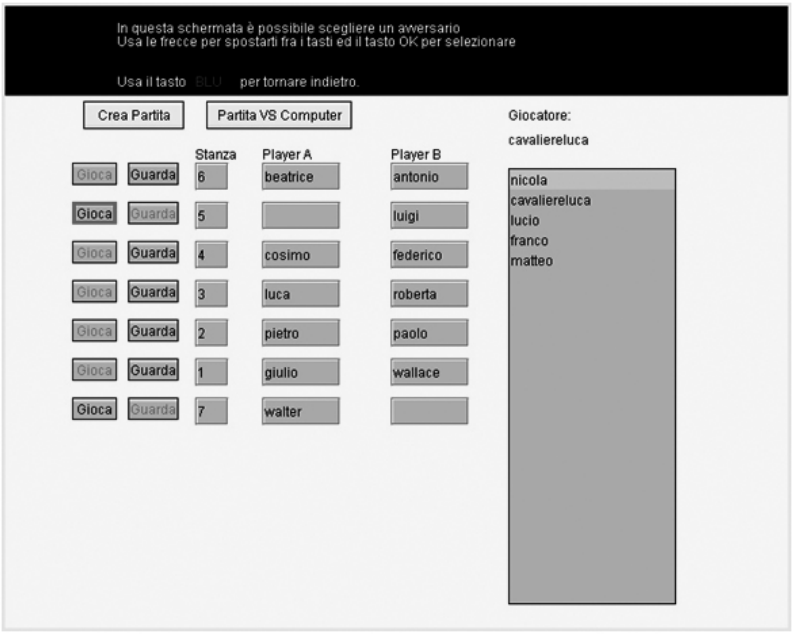

Figura 3.17 - Schermata di selezione delle stanze

Una volta effettuato il log-in, l'utente giunge alla schermata di selezione delle stanze (v. figura 3.17). In questa schermata è possibile avere la visione degli utenti collegati e delle stanze aperte. Da qui l'utente può:

- entrare in una stanza con un solo giocatore per iniziare una partita, spostandosi con i tasti freccia sul tasto *Gioca* relativo alla stanza scelta;

- entrare in una stanza nella quale c'è già una partita in corso per assistervi, selezionando il tasto *Guarda*;
- creare una nuova stanza, tramite il pulsante *Crea Partita*;
- giocare una partita contro un avversario virtuale (pulsante *Partita VS Computer*).

Scegliendo di accedere ad una stanza per giocare una partita, tramite il pulsante *Gioca*, l'utente giunge alla schermata presentata in figura 3.18.

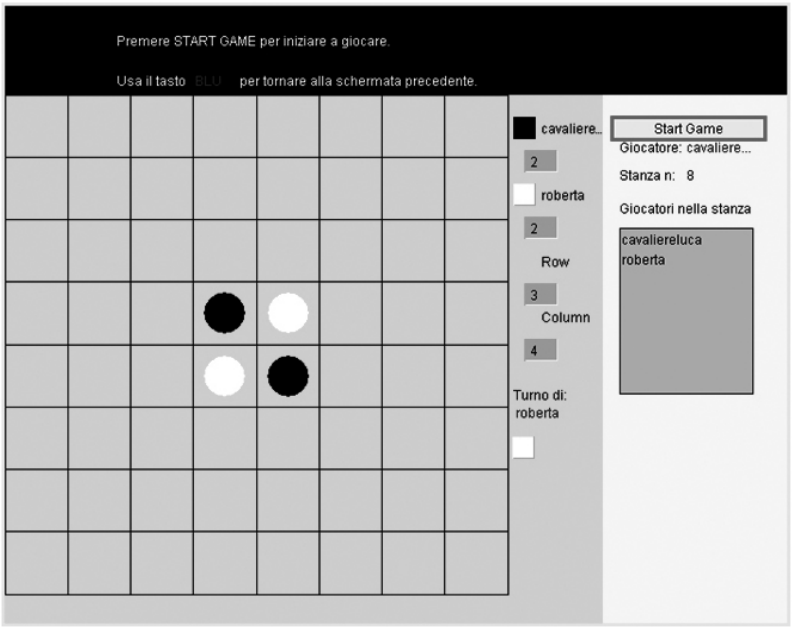

Figura 3.18 - Schermata di gioco

Dalla figura 3.18 è possibile notare che lo schermo è suddiviso in tre sezioni distinte: nella parte alta sono riportate informazioni sulla navigazione, sulla destra è presente la lista degli utenti presenti nella stanza ed il pulsante *Start Game* (che permette di comunicare all'avversario che si è pronti per giocare), mentre la parte centrale è dedicata allo stato del gioco e consente all'utente di operare le proprie mosse.

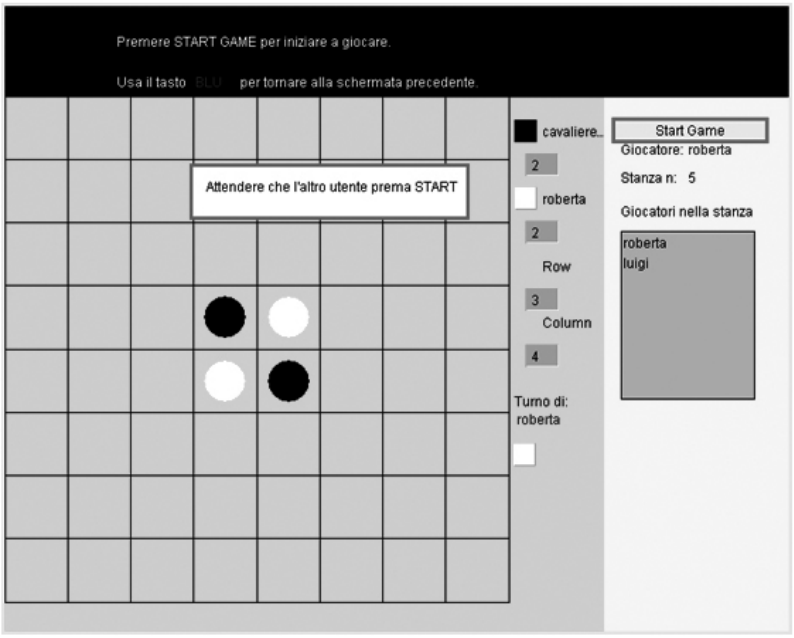

Figura 3.19 - Schermata di gioco dopo aver premuto il tasto *Start Game*

Una volta che entrambi i giocatori hanno premuto il tasto *Start Game*, la partita ha inizio: le mosse si effettuano spostandosi sulla scacchiera con i tasti "freccia" del telecomando e premendo il tasto di conferma per posizionare la pedina nella casella desiderata. Il cursore comunica, con la variazione del colore, le posizioni nelle quali è consentito posizionare la pedina (v. figura 3.20).

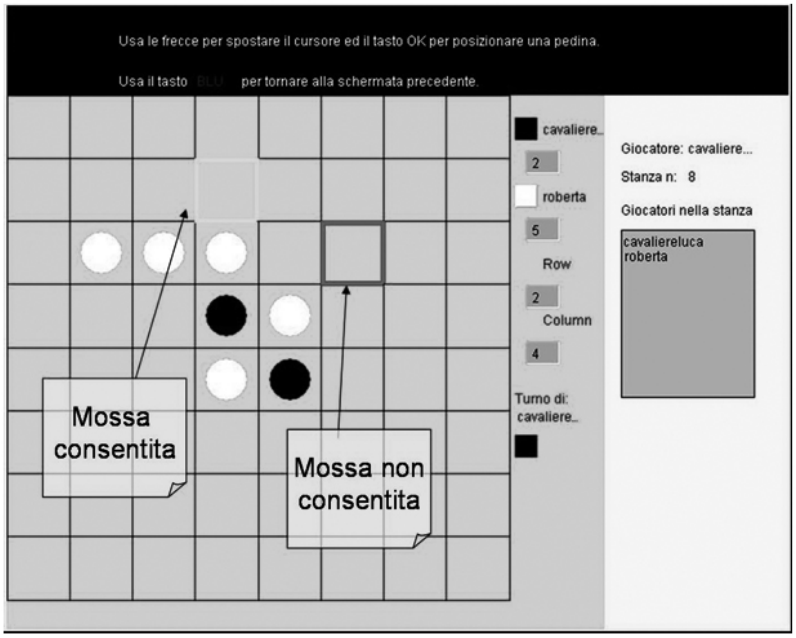

Figura 3.20 - Schermata di gioco con partita in corso

Prima di passare alla descrizione di alcuni ulteriori servizi basati sul framework MHP-VC, è interessante notare che nel community game sviluppato non entra in gioco, per il momento, l'agente chiamato User Profile Manager. Infatti la scelta dell'avversario al momento è a completa discrezione dell'utente e non viene fatta sulla base dell'analisi dei profili degli utenti attualmente connessi al servizio. Sarebbe interessante dotare il community game di alcune funzionalità di supporto all'utente per aiutarlo a scegliere un utente che presenta un'abilità di gioco paragonabile alla sua. Questo però implica che il sistema sia in grado di tenere traccia di ogni partita disputata dagli utenti, di analizzarne il risultato, e di aggiornare di conseguenza i profili degli utenti.

3.7.7 Sviluppo di Altri Servizi per Virtual Community

In questa ultima sezione vengono presentati alcuni dei servizi per virtual community che sono stati sviluppati successivamente ad Othello sfruttando le potenzialità del framework MHP-VC.

Sulla falsa riga di Othello sono stati implementati due ulteriori community game, chiamati "Trixlet" (il gioco del tris) e "Forza4". Il funzionamento delle applicazioni è molto simile all'esempio presentato nella sezione precedente, con l'aggiunta che in questo caso il sistema consente ai vari utenti di impostare alcune preferenze, come il proprio livello di gioco. Le preferenze fornite dagli utenti vengono memorizzate dagli agenti User Profile Manager, per far sì che un utente, nel momento in cui sceglie di giocare, possa ad esempio selezionare anche il livello di abilità dell'avversario.

Le figure 3.21 e 3.22 mostrano alcune schermate delle due applicazioni.

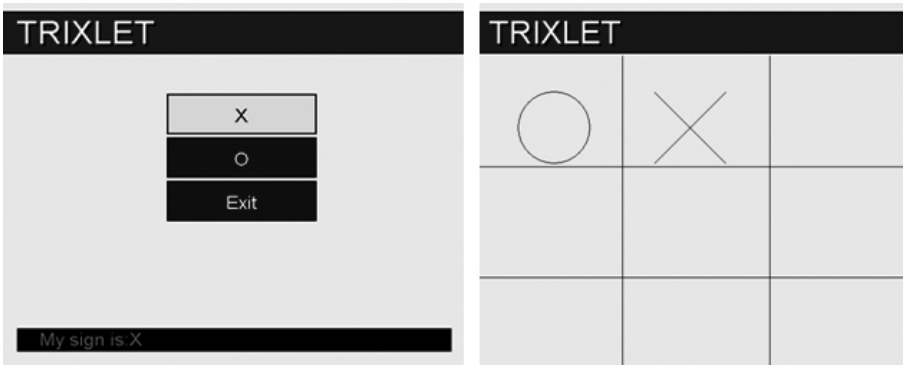

Figura 3.21 - Alcune schermate dell'applicazione Trixlet

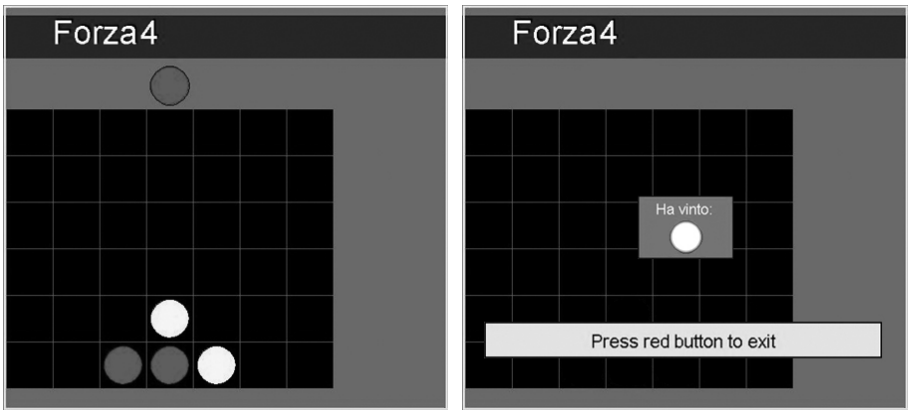

Figura 3.22 - Alcune schermate dell'applicazione Forza4

Un'altra applicazione sviluppata, che può essere considerata come un servizio di supporto ad altre applicazioni per virtual community, è una semplice chat che consente a due o più utente di interagire testualmente in tempo reale. Come negli altri esempi presentati, anche in questo caso abbiamo una schermata iniziale di autenticazione che richiede all'utente l'inserimento delle credenziali di accesso al servizio. La figura 3.23 presenta alcune schermate dell'applicazione.

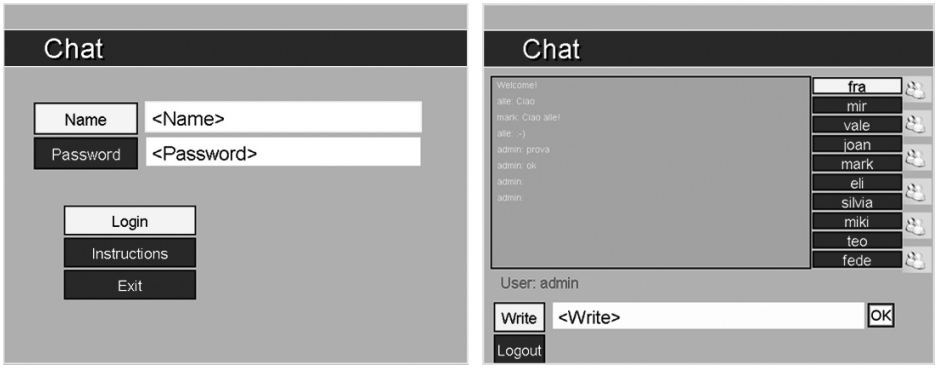

Figura 3.23 - Alcune schermate dell'applicazione Chat

Come è possibile notare dalla precedente figura, nell'applicazione "Chat" non è presente una tastiera su schermo. È infatti stato implementato un metodo alternativo per l'immissione di testo da parte dell'utente che permette la scrittura

dei caratteri tramite i tasti numerici del telecomando (una tecnica simile a quella adottata sui telefoni cellulari).

3.7.8 Sviluppi Futuri

Nelle precedenti sezioni è stato presentato un framework ad agenti finalizzato all'integrazione della tecnologia DTV-MHP con il concetto di virtual community, molto comune sul Web. Questo framework è stato progettato per offrire agli utenti MHP una nuova gamma di servizi per il supporto di attività collaborative e per consentire un'interazione in tempo reale fra diversi utenti. Con l'introduzione di questa nuova tipologia di servizi, possiamo affermare che le potenzialità della TV digitale possano raggiungere nuovi orizzonti in quanto, da un lato, il framework apre nuove vie di comunicazione e consente lo sviluppo di servizi MHP innovativi e, dall'altro lato, si rende possibile l'accesso ad applicazioni prima fruibili solo dagli utenti del Web senza richiedere particolari conoscenze informatiche, ma tramite il semplice utilizzo di un telecomando e di un decoder MHP.

L'implementazione del framework vede, lato server, una classica piattaforma ad agenti JADE collocata su un Web server e, lato client, una Xlet per ogni servizio caricata sul set-top box dell'utente. L'interazione fra le due componenti del framework avviene tramite lo scambio di messaggi ACL FIPA sul canale di ritorno del set-top box.

Gli sviluppi futuri della presente attività di ricerca sono essenzialmente due: da un lato sarebbe interessante chiudere l'anello presentato in figura 3.15, ossia fare in modo che il sistema multi-agente installato sul Web server sia anche in grado di comunicare con i dispositivi di broadcasting per aggiornare lo stato della Xlet direttamente tramite il flusso MPEG-2 inviato al decoder e, dall'altro lato, sarebbe sicuramente interessante proseguire nella progettazione e nella realizzazione di nuove applicazioni per incrementare ulteriormente le potenzialità della tecnologia DTV-MHP e fornire ai suoi utenti nuove tipologie di servizi al supporto di attività collaborative.

Conclusioni

L'attività di ricerca descritta nel presente lavoro di tesi si articola in tre filoni distinti, ognuno dei quali basato su una particolare tecnologia.

Inizialmente è stata studiata la tecnologia ad agenti, che ha consentito lo sviluppo della piattaforma RAVE e la conseguente implementazione dei sistemi RAP e CAFE, il primo al supporto di gruppi di programmatori Java ed il secondo incentrato su un approccio collaborativo al filtraggio ed alla classificazione di messaggi e-mail. I sistemi multi-agente rappresentano una costante la cui presenza ha caratterizzato l'intera attività di ricerca, che si è successivamente spostata in ambito multimediale, interessando le tecnologie VoIP e DTV-MHP.

La tecnologia Voice Over IP, grazie alle proprie potenzialità unite ai vantaggi economici dei quali i propri utenti possono beneficiare, ha riscosso un crescente interesse negli ultimi anni. Ad un'analisi preliminare che ha consentito di evidenziare pregi e difetti dei due maggiori standard VoIP, H.323 e SIP, è seguita una fase implementativa che ha portato allo sviluppo di alcuni interessanti strumenti come il sistema JadeSip, nel quale l'integrazione di due tecnologie eterogenee come SIP ed agenti JADE permette di fornire, da un lato, servizi di telefonia e messaggistica basati su SIP, e, dall'altro, alcune interessanti funzionalità di assistenza agli utenti.

L'ultima parte dell'attività di ricerca si è concentrata su un'altra tecnologia che si sta rapidamente diffondendo, la TV digitale interattiva basata sullo standard MHP. L'analisi dei progetti di ricerca che coinvolgono questa tecnologia ha permesso di evidenziare una totale assenza di aspetti collaborativi e di servizi che consentano un'interazione diretta tra gli utenti MHP. Da questa considerazione

hanno preso il via la progettazione ed il conseguente sviluppo del framework ad agenti MHP-VC, basato sull'integrazione della tecnologia DTV-MHP con il concetto di "virtual community", molto noto agli utenti della rete Internet, che possiamo in generale definire come uno spazio virtuale incentrato sull'interazione di più individui. L'idea alla base del progetto è stata quella di fornire alla fascia di utenza della TV digitale una serie di servizi evoluti basati appunto sull'interazione e sulla collaborazione diretta fra due o più utenti. Per testare le potenzialità del sistema sono state sviluppate alcune applicazioni tramite le quali è stato possibile testare praticamente il nuovo concetto di interattività introdotto, che si sposta dal classico paradigma MHP utente-applicazione verso una nuova ed interessante logica utente-utente o utente-comunità.

I diversi filoni di ricerca presentati, oltre ad essere accomunati a livello implementativo dall'utilizzo della tecnologia ad agenti, trovano un punto di incontro concettuale nell'aspetto collaborativo. In particolare, nei sistemi multi-agente descritti nel primo capitolo le caratteristiche di cooperazione e di collaborazione possono essere considerate sia un presupposto fondamentale richiesto alla comunità di utenti, sia il punto di arrivo che caratterizza i servizi offerti alla comunità stessa. D'altro canto, la tecnologia VoIP ed il framework DTV-MHP vedono nella collaborazione fra comunità di utenti il proprio scopo principale, fornendo servizi al supporto di attività collaborative.

Possiamo in conclusione definire l'attività di ricerca descritta nel presente lavoro di tesi come un percorso trasversale che, poggiandosi su tecnologie eterogenee, ha consentito lo sviluppo di una serie di interessanti ed innovativi strumenti al supporto di comunità di utenti impegnati in attività collaborative.

Bibliografia

[1] @lis TechNet Project. Web: http://www.alis-technet.org.

[2] A. McCallum, K. Nigam. *A comparison of event models for Naive Bayes text classification*. In M. Sahami (Ed) Proc. of AIII Workshop on Learning for Text Categorization, Madison, WI, pages 41-48, 1998.

[3] A. Vivacqua, H. Lieberman. *Agents to Assist in Finding Help*. In Proc. of ACM Conference on Human Factors in Computing Systems (CHI 2000), San Francisco, CA, 2000.

[4] C. Apte, F. Damerau, S.M. Weiss. *Automated learning of decision rules for text categorization*. ACM Trans. Inf. Syst., 12 (3), 233-251, 1994.

[5] C. Petrie. *Agent-based Engineering, the Web, and Intelligence*. IEEE Expert, 11(6), 1996.

[6] C. Romm, R. J. Clarke. *Virtual Community Research Themes: A Preliminary Draft for A Comprehensive Model*. In Proc. of Australasian Conference on Information Systems, 1995.

[7] Classifier4J Java Library. Web: http://classifier4j.sourceforge.net.

[8] Collaborator Project. Web: http://www.ist-collaborator.net.

[9] D. Lewis. *Feature selection and feature extraction for text categorization*. In Proc. of Workshop on Speech and Natural Language, Harriman, NY, USA, 1992.

[10] D. W. McDonald. *Evaluating expertise recommendations*. In Proc. of International ACM SIGGROUP Conference on Supporting Group Work, Boulder, CO, USA, 2001.

[11] Distributed Sender Blackhole List. Web: http://dsbl.org.

[12] DVB-MHP. Web: http://www.mhp.org.

[13] Ethereal Network Protocol Analyzer. Web: http://www.ethereal.com.

[14] F. Bellifemine, A. Poggi, G. Rimassa. *Developing Multi-agent Systems with a FIPA-compliant Agent Framework*. Software Practice and Experience, 31:103-128, 2001.

[15] F. Bergenti, A. Poggi, B. Burg, G. Caire. *Deploying FIPA-compliant Systems on Handheld Devices*. IEEE Internet Computing, 5(4):20-25, 2001.

[16] F. Bergenti, L. Lazzari, A. Poggi. *A Multi-Agent Framework to Join DTV and Virtual Communities.* In Proc. of Agent-based Computing for Enterprise Collaboration (ACEC) Workshop - WETICE 2006, Manchester, UK, 2006.

[17] F. Bergenti, L. Lazzari, A. Poggi. *Agent-Based Virtual Communities for Interactive Digital Television*. In Proc. of Workshop from Objects to Agents (WOA), Catania, Italy, 2006.

[18] FIPA (Foundation for Intelligent Physical Agents). Web: http://www.fipa.org.

[19] G. Salton. *Automatic Text Processing*. Addison-Wesley, 1989.

[20] H. Rheingold. *The Virtual Community: Homesteading on the Electronic Frontier*. Addison Wesley, 1993.

[21] I. Androutsopoulos, J. Koutsias, K.V. Chandrinos, G. Paliouras, D. Spyropoulos. *An evaluation of Naive Bayesian anti-spam filtering*. In Proc. of Workshop on Machine Learning in the New Information Age, Barcelona, Spain, 2000.

[22] I. Dagan, Y. Karov, D. Roth. *Mistake-driven learning in text categorization*. In C. Cardie, R. Weischedel (Eds.) Proc. of Conference on Empirical Methods in Natural Language Processing, Rhode Island, 1997.

[23] Iptel.org - IP Telecommunications Portal. Web: http://www.iptel.org.

[24] J. Goldbeck, J. Hendler. *Accuracy of metrics for inferring trust and reputation in semantic Web-based social networks*. In Proc. of International Conference on Knowledge Engineering and Knowledge Management, Northamptonshire, UK, 2004.

[25] J. Hagel, A. Armstrong. *Net Gain: Expanding Markets through Virtual Communities*. Harvard Business School Press, 1997.

[26] J. Heecheol, C. Petrie, M. R. Cutkosky. *JATLite: A Java Agent Infrastructure with Message Routing*. IEEE Internet Computing, 2000.

[27] J. Jung, G. Jo. *Collaborative Junk E-Mail Filtering Based on Multi-agent Systems*. Web Communication Technologies and Internet-Related Social Issues - HIS, 2003.

[28] J. Whatley. *Software Agents for Supporting Student Team Project Work*. In Proc. of International Conference on Enterprise System (ICEIS04), Porto, Portugal, 2004.

[29] JADE (Java Agent DEvelopment framework). Web: http://jade.tilab.com.

[30] JavaTV API. Web: http://java.sun.com/products/javatv.

[31] L. Lazzari, M. Mari, A. Poggi. *A Collaborative and Multi-Agent Approach to E-mail Filtering*. In Proc. of International Conference on Intelligent Agent Technology (IAT), Compiègne, France, 2005.

[32] L. Lazzari, M. Mari, A. Poggi. *A Collaborative and Multi-Agent System for E-mail Filtering and Classification*. In Proc. of

International Conference on Collaborative Computing (CollaborateCom 2005), San Jose, CA, USA, 2005.

[33] L. Lazzari, M. Mari, A. Negri, A. Poggi. *Agent-based Support for Open Communities*. In Proc. of Central and Eastern European Conference on Multi-Agent Systems (CEEMAS), Budapest, Hungary, 2005.

[34] L. Lazzari, M. Mari, A. Poggi. *CAFE - Collaborative Agents for Filtering E-mails*. In Proc. of Collaborative Peer to Peer Information Systems (COPS) Workshop - WETICE 2005, Linköping, Sweden, 2005.

[35] L. Lazzari, M. Mari, A. Negri, A. Poggi. *Support Remote Software Development in an Open Distributed Community*. In Proc. of Agent-Based Systems for Human Learning (ABSHL) Workshop - AAMAS 2005, Utrecht, The Netherlands, 2005.

[36] M. Hertzum, H.H.K. Andersen, V. Andersen, C.B. Hansen. *Trust in information sources: seeking information from people, documents, and virtual agents*. Interacting with Computers, pages 575-599 (14), 2002.

[37] M. Pazzani, D. Billsus. *Adaptive Web Site Agents*. Autonomous Agents and Multi-Agent Systems, 5, (2002) 205-218.

[38] M. Sahami, S. Dumais, D. Heckerman, E. Horvitz. *A Bayesian approach to filtering junk e-mail*. AAAI Workshop on Learning for Text Categorization, 1998.

[39] MD5 Hash Function Presentation Document. Web: http://theory.lcs.mit.edu/~rivest/rfc1321.txt.

[40] MjSip Java library. Web: http://www.mjsip.org.

[41] OpenH323 Project. Web: http://www.openh323.org.

[42] P. Resnick, I. Neophytos, S. Mitesh, P. Bergstrom, J. Riedl. *GroupLens: An open architecture for collaborative filtering of netnews*. In Proc. of Conference on Computer Supported Cooperative Work, Chapel Hill, NC, USA, 1994.

[43] Q. Jones, S. Rafaeli. *Time to Split, Virtually: 'Discourse Architecture' and 'Community Building' as means to Creating Vibrant Virtual Metropolises*. Int'l J. Electronic Commerce & Business Media, 10(4), 2000.

[44] S. N. Sanchez, E. Triantaphyllou, D. Kraft. *A feature mining based approach for the classification of text documents into disjoint classes*. Inf. Process. Manage., 38 (4), pages 583-604, 2002.

[45] SIP Communicator. Web: http://sip-communicator.org.

[46] T. Ishikawa, H. Matsuda, H. Takase. *Agent Supported Collaborative Learning Using Community Web Software*. In Proc. of International Conference on Computers in Education, Auckland, New Zealand, 2002.

[47] Vipul's Razor Spam Detection and Filtering Network. Web: http://razor.sourceforge.net.

[48] Vovida.org: Open-Source Communication. Web: http://www.vovida.org.

[49] W. Cohen. *Learning rules that classify e-mail*. In M.A. Hearst, H. Hirsh (Eds.) Proc. of AAAI Spring Symposium on Machine Learning in Information Access, Stanford, CA, USA, 1996.

[50] X. Liu, X. Zhang, L. Soh, J. Al-Jaroodi, H. Jiang. *I-MINDS: An Application of Multiagent System Intelligence to On-line Education*. In Proc. of IEEE International Conference on Systems, Man & Cybernetics, Washington, DC, USA, 2003.

[51] XleTView MHP Emulator. Web: http://xletview.sourceforge.net.