UNIVERSITÀ DEGLI STUDI DI TRENTO
Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea in Informatica

Tesi Specialistica

ToothAgent: un sistema multi-agente per il supporto di comunità mobili virtuali

Relatore:
Ing. Paolo Giorgini

Laureando:
Stefano Fante

Correlatore:
Dott. Luca Debiasi

Anno accademico 2004–2005

Indice

Introduzione

Nell'ultimo decennio abbiamo assistito ad uno sviluppo tecnologico profondo, caratterizzato tra l'altro da una diffusione massiccia di apparecchi legati alle comunicazioni quali i telefoni cellulari e, ultimamente, i computer palmari. Quello della comunicazione è stato il settore in cui il progresso è stato più evidente e la tendenza è quella di evolvere verso una situazione nella quale tutto deve poter comunicare. L'esplosione nella vendita di dispositivi di comunicazione mobile, cellulari su tutti, ha spinto la ricerca di nuovi metodi di comunicazione, non solo vocali ma anche digitali, per sfruttare appieno le loro potenzialità.

Oggi i cellulari non solo permettono la comunicazione tra persone via voce ma anche lo scambio di informazioni in formato elettronico; e grazie all'inserimento di unità di memoria sempre più capaci è possibile memorizzare grandi quantità di informazioni.

Gli utenti che utilizzano dei dispositivi mobili come telefoni cellulari o computer palmari possono formare delle comunità virtuali mobili, dove membri geograficamente co-localizzati possono collaborare e scambiarsi informazioni. Questo è motivato dal fatto che alla località geografica generalmente corrispondono interessi comuni e opportunità offerte da persone attive in una stessa area (ad esempio gli studenti di un'università possono essere interessati alla compravendita di testi usati per uno specifico

corso, oppure allo scambio di appunti riguardanti una lezione).

In questo contesto nasce l'idea per questo lavoro, cioè la progettazione e lo sviluppo di un sistema aperto e distribuito per il supporto dell'interazione di comunità (co-localizzate) virtuali mobili.

In letteratura si possono trovare alcune proposte relative al supporto di comunità virtuali mobili, ma quello che ancora manca è un'infrastrutura di supporto per l'interazione, la collaborazione e/o la competizione tra i membri della comunità virtuale. Solamente alcuni lavori introducono uno specifico ambiente collaborativo in cui i membri della comunità possono raggiungere delle forme di accordo attraverso la cooperazione.

Questo lavoro di tesi presenta un sistema multi-agente accessibile attraverso l'utilizzo di telefoni cellulari e PDA, in cui la tecnologia Bluetooth è adottata per connettere tali dispositivi e per riflettere la località degli utenti. Quello della "località" è infatti un concetto importante sul quale si è puntato. Essa sfrutta una caratteristica peculiare dei dispositivi mobili, ovvero quella di essere dove è fisicamente l'utente. Questo permette di realizzare un sistema in grado di fornire dei servizi contestualizzati, ai quali partecipano altri utenti che fisicamente frequentano quello stesso ambiente.

Esistono numerose piattaforme multi-agente che possono essere usate sui dispositivi mobili, tuttavia, tenendo conto delle ancora limitate risorse computazionali e di memoria che alcuni di essi possono avere, potrebbe risultare problematico eseguire un sistema multi-agente su un dispositivo di tal genere. Una possibile soluzione è quella di lanciare gli agenti anziché sul proprio cellulare/PDA, su una piattaforma multi-agente in un host esterno.

Questo lavoro presenta un'architettura generale basata proprio su quest'ultima opzione. Viene proposto l'utilizzo di server indipendenti dove possano essere installate piattaforme multi-agente e dove gli agenti possano

agire in base ai desideri dei loro utenti. Ogni server può proporre uno o più servizi specifici relativi alla propria area geografica e gli utenti possono contattare i loro agenti personali utilizzando i loro telefoni cellulari dotati di tecnologie Java (per la natura del programma sviluppato) e Bluetooth.

Il sistema è indipendente dal dominio (non dipende dai servizi specifici offerti dal server) e dalla tecnologia multi-agente adottata (possono essere utilizzate differenti tecnologie su ogni server).

Il lavoro si articola come di seguito: il primo capitolo tratterà le comunità virtuali mobili e le tecnologie che le supportano; verranno inoltre descritti i tool esistenti e discussi i loro limiti. Il secondo capitolo introdurrà *ToothAgent*, il sistema sviluppato; verranno quindi descritte la sua architettura generale e le sue componenti, offrendo alcuni possibili scenari in cui il software prodotto potrebbe risultare utile agli utenti e suggerendo alcune tipologie di servizio. Verrà trattato inoltre il framework di Cultura Implicita, inserito nel sistema al fine di migliorarne la qualità e le prestazioni. Il terzo capitolo descriverà invece l'implementazione e i test effettuati. Nell'ultimo capitolo verranno infine tirate le conclusioni e proposti gli ulteriori possibili sviluppi futuri del sistema.

Capitolo 1

Comunità virtuali mobili e tecnologie di supporto

In questo capitolo si introdurrà il concetto di comunità virtuale e si analizzeranno in particolare le comunità virtuali mobili. Verrà inoltre fatta una panoramica sugli agenti e sui sistemi multi-agente, coinvolgendo la comunicazione Bluetooth e illustrando lo stato dell'arte dei sistemi di supporto per comunità virtuali che utilizzano queste tecnologie.

1.1 Le comunità virtuali mobili

Con lo sviluppo delle tecnologie per la comunicazione mobile, intesa non solo come comunicazione vocale ma anche come scambio di informazioni (sms, mms, foto e in generale ogni tipo di file) è nata l'idea di formare dei gruppi entro i quali le persone potessero interagire, confrontarsi, comunicare. A questi gruppi di persone è stato dato il nome di comunità virtuali poichè i membri che le compongono possono anche non conoscersi e non avere rapporti di alcun tipo al di là di quelli virtuali appunto. La particolarità di

queste comunità sta nel fatto che hanno un carattere mobile poiché utilizzano dispositivi come telefoni cellulari, PDA, notebook, smartphone, ecc. e non sono legate al luogo in cui operano, come l'ufficio o il posto di lavoro: esse vengono quindi identificate con l'aggettivo che le contraddistingue dalle classiche comunità virtuali e vengono quindi chiamate comunità virtuali mobili [14].

Le comunità virtuali in genere (Virtual Community - VC) hanno la caratteristica di privilegiare, nel rapporto tra due o più individui che vivono anche in paesi diversi, la condivisione di interessi comuni; ciò avviene in virtù dell'abbattimento delle barriere socio-culturali che possono creare, nelle comunità reali, le distanze comunicative.

La comunicazione virtuale permette inoltre il gioco di identità e lo scambio dei ruoli; come ciò possa influire sulla psicologia di un individuo rimane una riflessione ancora poco esplorata in campo psico-sociologico; comunque, l'abilità di mascherare la propria identità può produrre effetti diversi, negativi e positivi.

Un altro vantaggio delle comunità virtuali è la sensibile diminuzione delle inibizioni da parte di chi ne fa parte. Le persone più timide che non riescono ad intervenire in un gruppo e hanno difficoltà a partecipare alla conversazione, se hanno bisogno di tempo per riflettere, nelle comunità virtuali hanno la possibilità di essere ascoltate in un modo impossibile altrove. Questo diminuisce l'inibizione che impedisce, normalmente, la partecipazione alla conversazione.

Le comunità virtuali forniscono informazioni a cui è possibile accedere ovunque; tali informazioni tuttavia, per essere accessibili ed usabili in ogni luogo e in qualunque momento, necessitano di una tecnologia di comunicazione appropriata che diviene l'elemento cruciale del sistema. Dalla

combinazione delle tradizionali tecnologie internet con le nuove capacità delle reti mobili, si ottiene un approccio per far fronte alla sfida di "accedere e connettersi in ogni momento, da ogni luogo ed in ogni situazione". La visione delle comunità virtuali mobili implica un ponte fra la tecnologia e i requisiti degli utenti ed assume - per esempio - che la sincronizzazione fra differenti dispositivi permetta interazioni che non creano problemi e che non necessitano di giunzioni. In breve, le comunità mobili virtuali hanno l'enorme potenziale di servire in qualunque momento i desideri degli utenti.

Howard Rheingold, conosciuto nella comunità scientifica soprattutto per aver inventato l'acronimo "comunità virtuali", ha recentemente coniato anche il termine Smart Mobs [19]; con tale termine ha voluto riferirsi ad un gruppo di persone facenti parte delle più moderne comunità high-tech "intelligenti", in cui le persone utilizzano le molteplici tecnologie oggi disponibili per organizzarsi e coordinarsi in azioni collettive di vario genere. Fra le altre anche la tecnologia wireless, da intendersi non solo come telefonia cellulare, ma anche pagers (tastiere elettroniche per scrivere sms, molto diffuse soprattutto in America), personal digital assistants (minicomputer tascabili), micro-PC wearables (indossabili). Gli Smart Mobs usano le connessioni wireless per organizzarsi in azioni comuni come fossero un esercito virtuale equipaggiato di gadget altamente tecnologici per la comunicazione a distanza. Tutti questi gadget permettono loro di agire in modo coordinato anche senza conoscersi gli uni con gli altri.

Tutto ciò che gira attorno agli Smart Mobs è in rapidissima evoluzione: i dispositivi intelligenti sempre più piccoli e potenti, le imprese del mercato delle telecomunicazioni che costruiscono sofisticate tecnologie high-tech alla portata delle masse, le regole di comportamento digitale da seguire per usare questi mezzi di comunicazione, ecc. Gli stessi conflitti politici e

sociali per il controllo delle tecnologie pervasive (ubiquitous computing) getteranno le basi per la vita futura. Le comunità virtuali wireless (ad esempio www.thefeature.com), i webloggers (fenomeno internet molto in voga negli ultimi tempi), i compratori e venditori presenti nelle aste on line su siti come eBay, sono esempi di contesti dove gli Smart Mobs si ritrovano virtualmente tra loro.

Una comunità virtuale può tuttavia essere costituita in un qualsiasi ambito: basti pensare ad un'università, in cui gli studenti con interessi sostanzialmente comuni potrebbero formare dei gruppi per la discussione di argomenti specifici, per lo scambio di informazioni o appunti inerenti i corsi universitari, o ancora per l'organizzazione di eventi culturali o sportivi. Oltre all'ormai tradizionale uso di internet, delle e-mail, dei blog e dei forum, potrebbero affiancare l'utilizzo dei dispositivi mobili e continuerebbero ad essere tra loro collegati e connessi, potendosi confrontare e condividendo delle risorse utili all'intera comunità: tutto questo anche senza l'obbligo di conoscersi, incontrarsi, scambiarsi informazioni personali.

Oltre a incontri virtuali, gli Smart Mobs usano le tecnologie per organizzarsi anche fisicamente. Howard Rheingold ha chiamato questo fenomeno *swarming tactic* (chiamata a raccolta di una folla di persone nel minor tempo possibile), che ad esempio migliaia di cittadini filippini hanno adottato per cacciare il Presidente Estrada nel 2000, usando gli sms per radunarsi in piazze e cortei.

Tecnologie della comunicazione sempre più pervasive rappresentano risorse estremamente positive per comunicare in modo semplice e sempre più efficace. Al contempo potrebbero però rivelarsi potenzialmente distruttive se utilizzate per semplificare l'organizzazione di azioni contro altre persone. Si pensi alle analogie e alle differenze di casi come le marce della pace

e gli attacchi dell'11 settembre. In entrambi i casi la tecnologia della comunicazione è stata utilizzata per organizzare incontri e azioni di gruppi di persone. Oggi, le tecnologie che facilitano la cooperazione possono essere impiegate anche per scopi immorali, per esempio, diffondere istruzioni su come costruire la bomba nucleare o mine antiuomo. Gli analisti della RAND Corporation hanno scoperto che la nuova mafia russa e i narcotrafficanti colombiani attuano la cosiddetta "guerra tecnologica" per realizzare i loro traffici illeciti, combinando comunicazioni satellitari e utilizzando siti internet per organizzare le operazioni.

Tuttavia, escludendo questi scenari negativi, è facilmente prevedibile che in futuro molti paesi inventeranno nuovi modi di comunicare, interagire, fare affari (così come è accaduto in Giappone negli ultimi anni), secondo logiche che gli stessi inventori delle tecnologie Smart Mobs non avevano neanche immaginato.

Diviene quindi chiara la necessità di supportare queste comunità al fine di agevolare la comunicazione interna dei membri che le compongono. Il tema è stato trattato in letteratura ed attualmente vengono proposti degli applicativi basati sulla comunicazione mobile via GPS, IR, Bluetooth, GPRS, Wi-Fi, ecc. Il cuore del sistema, al di là della metodologia di comunicazione, sta però nella piattaforma che gestisce gli utenti e che mira al soddisfacimento dei loro desideri, sia che essi ricerchino semplici informazioni statiche, sia che necessitino di complicate interazioni e contrattazioni per ottenete il risultato finale. Il software che si occupa di questo, deve infatti garantire all'utente i risultati che si attende di ricevere.

Un esempio di progetto di ricerca per il supporto di utenti mobili è dato da *PortableCicero* [11]: esso utilizza dei raggi IR posti all'entrata di ogni sala di un museo per individuare il posto esatto in cui si trova l'utente

per poi poter fornire ad esso tutte le informazioni di cui necessita. Grazie a delle mappe precaricate sul PDA in possesso dell'utente, il sistema può mostrare la piantina della stanza; e grazie ad un'interfaccia grafica, il turista può scegliere l'opera d'arte per cui desidera informazioni. Sebbene molto valido, PortableCicero presenta però l'inconveniente di richiedere una certa dimestichezza con i computer palmari.

MIA (Mobile Information Agents) [4] è un altro esempio di sistema che offre informazioni personalizzate e localizzate agli utenti via dispositivo mobile. Lo scopo di MIA è sviluppare un sistema informativo intelligente che invii informazioni di rilevanza locale dal World Wide Web ai cellulari. Un sistema di questo tipo può essere utile per esempio ad una persona che arriva in una nuova città e potrebbe essere interessata a conoscere dei ristoranti in cui sia possibile cenare: MIA si può occupare del recupero di queste informazioni, filtrando i risultati sulla base delle preferenze impostate e le può offrire all'utente, indicando magari anche la strada per arrivare al ristorante. Come il caso precedente, anche MIA è un sistema client-server che però sfrutta il ricevitore GPS anziché i raggi infrarossi per individuare la posizione del cliente. È evidente che gli attuali telefoni cellulari non integrano ancora questa tecnologia, e questo rende di fatto MIA utilizzabile solo attraverso quei pochi PDA che già lo fanno o solo quando l'utente si trova in auto, dove ha installato il ricevitore GPS. Gli sviluppatori di MIA stanno ora lavorando sulla soluzione aggiuntiva di rendere il sistema compatibile con tutti i telefoni cellulari compatibili con WAP. In questo caso la posizione dell'utente sarebbe determinata (anche se con molta meno precisione) osservando la cella in cui si trova. La comunicazione via WAP presenterebbe però lo svantaggio di non essere gratuita: gli utenti sarebbero infatti costretti a pagare le informazioni ricevute in base ai loro costi di

connessione alle rete e alla quantità di dati scaricati.

Nel campo del supporto informativo ai turisti esistono ovviamente altri progetti di ricerca che sfruttano la località per fornire aiuto e indicazioni; è il caso di alcuni progetti *IST*. Essi si concentrano sul problema dell'assistenza ad un turista durante le sua visita alla città: forniscono informazioni sui monumenti di interesse, sulla chiese, sulla strade, sugli eventi, ecc. ed aiutano il turista alla pianificazione dei propri itinerari. Anche in questi casi le tecnologie adottate per la comunicazione possono essere tanto il Bluetooth quanto il GPS: in quest'ultimo caso gli svantaggi sarebbero gli stessi sopra riportati.

In altri contesti, differenti da quello delle semplici informazioni su come raggiungere una meta o sui dettagli di un'opera d'arte, l'utente si aspetta che il prorio software esegua del lavoro per lui in modo autonomo, proattivo e/o propositivo, contrattando, collaborando o entrando in competizione con software di altri utenti. In questi casi si potrebbe implementare un'altra tipologia di sistema, utilizzando la cosiddetta programmazione ad agenti. Seguendo questo paradigma, sulla piattaforma multi-agente vivrebbero ed opererebbero delle entità software (dette appunto agenti) che agirebbero in modo trasparente fornendo all'utente solamente i risultati finali.

1.2 Agenti e sistemi multi-agente

La delegazione dei compiti e l'intelligenza implicano il bisogno, o meglio la necessità, di costruire sistemi di computer che possano agire effettivamente in base ai nostri comportamenti: questo significa che i computer devono essere in grado di agire indipendentemente e nella maniera che meglio interseca i nostri desideri e i nostri interessi, interagendo con altri umani o con altri sistemi. Poiché l'interconnessione e la distribuzione sono diventati il

tema dominante della Computer Science, essi, con la necessità dei sistemi di soddisfare al meglio i nostri desideri, hanno portato allo sviluppo di software che possono cooperare e accordarsi (e perché no, competere) con altri sistemi che hanno differenti interessi (perché differenti sono gli interessi delle persone per cui lavorano). Questo problema non è stato affrontato che di recente e ha portato alla creazione di un nuovo campo della Computer Science: i sistemi multi-agente.
Esistono varie definizioni (controverse) di "agente"; le più comuni e utilizzate sono tuttavia le seguenti:

1. Un agente è un sistema capace di azioni indipendenti nell'interesse del proprio proprietario (estrapolando i desideri che devono essere soddisfatti per raggiungere gli obiettivi piuttosto che chiederli ripetutamente) [16].

2. Un agente è un'entità software che funziona continuamente ed autonomamente in un particolare ambiente, spesso abitato da altri agenti e processi [12].

Altre definizioni concernono invece la conoscenza, l'apprendimento, la capacità deduttiva, ... che sono equamente importanti:

1. Un agente è un'entità software che può possedere alcuni dei seguenti attributi:

    - Reattività (intuizione ed azione selettiva)
    - Autonomia (orientamento all'obiettivo, proattività e comportamenti auto eseguiti)
    - Collaborazione (lavoro con altri agenti ed entità per raggiungere un obiettivo comune)

- Abilità di comunicazione in rapporto al livello di conoscenza (comunicazione con altre entità utilizzando un linguaggio comune)
- Capacità deduttiva (esecuzione di task astratti utilizzando propri modelli, apprendendo situazioni o comportamenti di altri agenti)
- Continuità temporale (persistenza dello stato e della personalità)
- Personalità (manifestazione di attributi di un agente credibile)
- Adattività (apprendimento e miglioramento con l'esperienza)
- Mobilità (migrazione da un host ad un altro in modo autonomo)

2. Un agente è un'entità software che esibisce (un sottoinsieme del) le seguenti caratteristiche:

   - Informazione (capacità cognitive, interazione, cooperazione, coordinazione e competizione con altre persone, agenti e processi)
   - Apprendimento (capacità di imparare autonomamente)
   - Autonomia (proattività)
   - Comunicazione
   - Mobilità (mobilità fisica o software)

Tuttavia, al di là delle definizioni, lo sviluppo di sistemi ad agenti è stato finora dominato da linee guida informali, euristiche ed ispirazioni anziché principi formali e tecniche ingegneristiche ben definite.

Un sistema multi-agente (Multi-Agent System, MAS) è un sistema composto da un certo numero di agenti, capaci di mutua interazione. L'interazione può essere in forma di passaggio di messaggi o produzione di cambiamenti di stato nel loro comune ambiente. L'esatta natura degli agenti è tuttavia argomento di qualche controversia. Essi sono solitamente

detti essere autonomi, d'altro canto, in pratica, tutti gli agenti sono sotto la supervisione umana. Inoltre, più importante è l'attività dell'agente per gli umani, più supervisione essi devono ricevere. Infatti l'autonomia completa è raramente desiderata. Si ha invece bisogno di sistemi interdipendenti.

Ai sistemi multi-agente può essere richiesta l'inclusione di agenti umani; le organizzazioni umane e le società in genere possono essere considerate come un esempio di sistema multi-agente. I MAS possono manifestare auto-organizzazioni e comportamenti complessi anche quando le strategie individuali di tutti gli agenti sono semplici.

Alcuni sistemi richiedono quindi un livello di autonomia e dinamicità che, riteniamo, solo un sistema multi-agente possa offrire.

In letteratura si possono trovare alcuni tool basati su piattafome multi-agente e comunicazione wireless: è il caso di sistemi come:

- *KORE* [17], un sistema multi-agente in cui una guida elettronica personale fornisce al visitatore di un museo provvisto di un dispositivo mobile abilitato Java informazioni riguardanti le opere d'arte che sta guardando. La posizione dell'utente viene determinata sfruttando le caratteristiche della connessione Bluetooth e le informazioni vengono filtrate in base alle preferenze dell'utente stesso.

- *MobiAgent* [20], un framework basato sugli agenti che permette agli utenti di accedere a vari tipi di servizio (dalla ricerca web al controllo di applicazioni remote) direttamente usando i propri telefoni cellulari o PDA. Quando un utente invia la richiesta per uno specifico servizio (tramite http e non tramite Bluetooth, dribblando così il concetto di località spesso associato a questo tipo di sistemi), un agente inizia a lavorare per lui in base ai suoi desideri e alle sue intenzioni. L'agente può a quel punto disconnettersi dalla rete senza che il suo agente ne

risenta: esso continuerà infatti il proprio lavoro fino a quando l'avrà terminato. L'utente verrà a quel punto avvisato tramite un sms e potrà decidere se e quando riconnettersi per scaricare i risultati.

Quello che è realmente mancante nei sistemi sopra riportati è però la possibilità di supportare svariati servizi che coinvolgano l'interazione, la collaborazione e la competizione fra gli agenti membri della comunità virtuale. Solo poche proposte in letteratura introducono un ambiente specificatamente collaborativo dove gli agenti possono agire per conto dei propri utenti e in base ai loro desideri. È il caso di sistemi come quelli presentati nei seguenti lavori:

- [21] descrive un sistema multi-agente per la gestione di un'agenda in cui vi sono degli agenti che, lavorando sui PC o sui PDA, si occupano dello scheduling e assistono i loro utenti nell'impostazione degli incontri e nel prendere degli impegni negoziando con gli altri agenti per arrivare ad un comune accordo. In questo progetto si sono utilizzate due tipologie di framework: JADE, presente sul PC e LEAP [9], installata invece sui PDA.

- [22] descrive un progetto simile, in cui però non è previsto l'uso di dispositivi mobili, ma solamente di PC collagati in rete e di piattaforme multi-agente sviluppate con JADE.

- [18] presenta *ADOMO* (ADvertising On MObile Phones), un sistema in cui gli agenti presenti sul cellulare/PDA negoziano e stabiliscono contratti con agenti di tipo commerciale presenti nelle vicinanze utilizzando una connessione Bluetooth. L'oggetto delle negoziazioni e dei contratti sono pubblicità commerciali che devono essere mostrate sul display del dispositivo mobile in cambio di qualche piccola

ricompensa/beneficio. L'idea di base è quella per cui alcuni negozi, ristoranti, ecc. installano l'applicativo sui propri computer e attivano i propri agenti. Quando un utente interessato ad una categoria specifica di pubblicità passa nelle vicinanze del server in questione, viene avvisato tramite un messaggio delle offerte, delle promozioni, del menu (se si tratta di un ristorante o di una pizzeria) o di qualsiasi altra notizia il commerciante/ristoratore voglia offrire; tutto questo in cambio di uno sconto, una piccola ricompensa o un trattamento speciale. A differenza dell'applicazione precedente, in ADOMO non esiste però la competizione degli agenti, ma solo una loro collaborazione, per ottenere un risultato che possa avvantaggiare entrambi.

Con l'utilizzo del paradigma ad agenti, come del resto anche utilizzando altri approcci, l'utente ha la necessità di osservare il lavoro per lui prodotto dal sul agente personale (d'ora in poi personal agent). Questo lavoro può chiaramente essere osservato solo attraverso dei messaggi inviati dall'agente (o da una qualche altra entità facente parte del sistema) all'utente. Quando si lavora a livello di simulazione o comunque lavorando sul proprio PC, le informazioni possono essere tranquillamente osservate a video; quando invece si parla si Smart Mobs e di utenti mobili, risulta chiara la necessità di far giungere le informazioni a coloro che le desiderano in altro modo. Ed ecco quindi l'utilizzo di altre tecnologie per le comunicazioni mobili come quelle citate in precedenza: il WAP, il GPS e il Bluetooth. Proprio su quest'ultima si focalizza la nostra attenzione: il Bluetooth infatti, per la sua portata limitata, consente lo scambio di messaggi solo a quegli utenti che si trovano nelle vicinanze di un punto di accesso (access point) attribuendo al sistema un carattere prettamente localizzato e permettendo quindi di fornire

servizi ad hoc, specifici per ogni luogo e per ogni tipologia di utente.

1.3 La comunicazione Bluetooth

Lo scambio di dati e di informazioni fra dispositivi mobili può essere effettuato in vari modi, con differenti costi di utilizzo e prestazioni. Uno di questi modi è l'utilizzo di una tecnologia wireless chiamata Bluetooth. Questa tecnologia si colloca in una posizione privilegiata rispetto ad altre tecnologie come il WAP o il GPS per vari motivi:

- innanzitutto per il suo carattere tipicamente localizzato, essendo il suo raggio di azione limitato;
- secondo per i suoi costi di connessione, nulli, a differenza del WAP per cui si deve pagare un prezzo che varia a seconda della compagnia telefonica e della quantità dati che vengono scaricati;
- terzo, ma non per questo ultimo, per la grande diffusione (a bassi costi) che il Bluetooth sta avendo in questi ultimissimi anni, a differenza del GPS che non è ancora integrato nei cellulari e lo è solo in pochissimi PDA.

In questa sezione verrà data una breve introduzione della tecnologia Bluetooth fornendo qualche dato e spiegando (in maniera non esaustiva) i protocolli di comunicazione.

Cenni storici

Le prime ricerche sulla tecnologia Bluetooth partono nel 1994, quando la Ericsson Mobile Communication intraprende una serie di studi allo scopo di trovare delle alternative wireless per il collegamento tra telefoni e accessori

(es. auricolari). Lo studio riguarda un collegamento radio che, non essendo direzionale, non necessita della cosiddetta "line of sight", cioè della visibilità diretta, ed ha quindi degli evidenti vantaggi rispetto alle tecnologie infrarossi precedentemente utilizzate per collegare tra loro cellulari e dispositivi vari. Nel febbraio 1998 nasce il SIG (Special Interest Group), che è un gruppo di compagnie leader del settore delle telecomunicazioni e dell'elettronica, tra cui Ericsson, Nokia, Intel, IBM e Toshiba, che lavorano assieme per promuovere e definire le specifiche Bluetooth. In seguito entrano a far parte del SIG anche Microsoft, Lucent, 3Com e Motorola; al giorno d'oggi il SIG comprende più di 2000 compagnie. Nel luglio 1999 viene pubblicata la versione 1.0 delle specifiche Bluetooth, che viene seguita nel dicembre 1999 dalla versione 1.0b. Nel febbraio 2001 esce la versione 1.1, mentre a novembre del 2003 viene rilasciata la versione 1.2.

La più grossa novità introdotta dalla Bluetooth Core Specification Version 1.2 è rappresentata dall'Adaptive Frequency Hopping (AFH), una tecnica pensata per ridurre al minimo le interferenze fra le varie tecnologie wireless che condividono la gamma di frequenze dei 2,4 GHz. L'AFH si preoccupa di analizzare dinamicamente lo spettro in cerca delle frequenze già occupate da altri dispositivi, come i telefoni cordless e i device Wi-Fi (802.11b/g), e utilizzare solo quelle disponibili: secondo gli ingegneri che l'hanno sviluppato questo può tradursi, in ambienti con molte interferenze, in un sensibile incremento delle prestazioni.

L'altra miglioria promessa dalle nuove specifiche 1.2 riguarda la velocità di instaurazione della connessione fra periferiche, un aspetto che nel 2002 il Bluetooth SIG si ripromise di affinare con il varo del progetto "Five Minute Ready" (pronto in cinque minuti). La nuova versione di Bluetooth include diversi nuovi metodi per la ricerca dei device che, attraverso l'adozione di

algoritmi ottimizzati, riducono i tempi necessari affinché due o più dispositivi "si vedano" e siano pronti a scambiarsi dati: i progettisti hanno fatto in modo che i benefici apportati da queste nuove tecniche possano essere parzialmente sfruttati anche nel caso in cui solo il dispositivo ricevente sia aggiornato alla versione 1.2 dello standard.

Attualmente, è ancora in fase di definizione la versione 2.0; tuttavia un primo aggiornamento di questa tecnologia si è avuto all'inizio del 2005, con l'introduzione sul mercato dei primi dispositivi rispondenti alle specifiche Bluetooth 2.0 con EDR (Enhanced Data Rate).

Entro la fine del 2006 si calcola che il 75% del cellulari implementeranno la tecnologia Bluetooth. La compagnia di ricerche di mercato In-Stat ha infatti condotto un'analisi secondo la quale la diffusione dei dispositivi equipaggiati con tecnologia Bluetooth assisterà ad una consistente impennata a partire dalla fine dell'anno in corso, grazie alla crescente popolarità di soluzioni wireless a corto raggio per voce e dati.

Brian O'Rourke, analista di In-Stat, ha dichiarato: "I telefoni cellulari rappresentano il motore della popolarità della tecnologia Bluetooth. Grazie ad essi tale tecnologia wireless è riuscita a diffondersi attraverso altri prodotti come, notebook, auricolari e cuffie, automobili e lettori portatili di musica digital".

Le previsioni di consegne per l'anno in corso sono di 316 milioni di unità, un numero che dovrebbe crescere fino a quota 866 milioni nel 2009. Secondo l'analisi, comunque, la tecnologia Bluetooth dovrà continuamente tenersi al passo per fronteggiare altre soluzioni che sono potenzialmente caratterizzate dallo stesso ambito di impiego, come ad esempio Wireless USB, Wireless 1394 e Wi-Fi.

Generalità

Bluetooth è una tecnologia radio studiata appositamente per trasmissioni a corto raggio (tipicamente 10 m) ed è caratterizzata da un bassissimo consumo di potenza, da un basso costo e da una bassa complessità. Queste caratteristiche consentono a Bluetooth di proporsi come la tecnologia del futuro nel campo delle comunicazioni wireless per apparecchi di piccole dimensioni alimentati a batteria. Inoltre è studiata anche per connettere e fare interagire tra di loro dispositivi diversi (telefoni, stampanti, notebook, PDA, impianti HiFi, TV, PC, cellulari, elettrodomestici, ecc). Bluetooth permette anche di accedere a reti locali (LAN) tramite dei dispositivi denominati Access Point, nonché alla rete pubblica PSTN (Public Switched Telephon Network) e alla rete di telefonia mobile.

Lo standard Bluetooth consente ai device di connettersi e comunicare tra loro in una regione limitata attraverso una rete ad hoc chiamata piconet, che è costituita da un massimo di otto dispositivi attivi, di cui uno è il master, ossia colui che inizia lo scambio di dati, e gli altri sette sono gli slave, che funzionano in risposta al master, e non hanno altri collegamenti oltre a quello col master. Inoltre è possibile avere fino a 200 dispositivi in modalità “park”, cioè che non possono essere attivi nello scambio di dati col master, ma che restano sincronizzati con esso.

Caratteristica importante dei dispositivi Bluetooth è la loro capacità di creare e riconfigurare dinamicamente queste piconet senza richiedere l’intervento umano. Infatti se un dispositivo entra o esce dalla zona coperta dal master viene riconosciuto o tolto dalla rete automaticamente. Questo grazie all’operazione di inquiry, attraverso la quale i dispositivi Bluetooth periodicamente sono in grado di scoprire l’esistenza di altri nelle vicinanze. Questa possibilità di configurazione automatica permette, ad esempio, di

sincronizzare i dati di un PC portatile e un PDA semplicemente avvicinando i due apparecchi oppure di passare automaticamente al vivavoce quando si entra in auto parlando al cellulare, fino al caso di un operatore industriale che con un palmare può muoversi all'interno di una fabbrica e monitorare e configurare i vari macchinari in maniera veloce ed efficiente. Tutto questo è possibile grazie al "Service Discovery Protocol" (SDP), che permette ad un dispositivo Bluetooth di determinare quali sono i servizi che gli altri apparecchi presenti nella piconet mettono a disposizione. Tale protocollo può fungere sia da server (ossia può essere interrogato da un altro dispositivo e rispondere con i propri servizi) sia da client (interrogando gli altri dispositivi) e ogni apparecchio dispone delle informazioni relative ai servizi di cui è capace e dei protocolli supportati: altri apparati potranno fare uso di queste informazioni per determinare le possibilità di interazione con i nodi della piconet. Questo è necessario perché, naturalmente, una stampante Bluetooth non offre le stesse possibilità di un PDA o di un'auricolare, pertanto occorre che ogni nodo conosca le funzioni e le possibilità di ogni altro nodo della rete. Per fare un esempio concreto, se un telefonino Bluetooth vuole trasferire un messaggio di testo a un PDA, potrà interrogare quest'ultimo per sapere se è dotato di funzionalità e-mail, o se è in grado di ricevere un testo in altro modo. Quando un dispositivo si inserisce per la prima volta in una piconet inoltre, effettuerà una "scansione" di tutti i nodi presenti per capire come può interagire con essi.

Altra caratteristica importante dei dispositivi Bluetooth è quella di poter essere presenti in più di una piconet contemporaneamente, e di poter addirittura funzionare da master in una piconet e da slave nell'altra. Un gruppo di piconet legate tra loro viene chiamata scatternet.

Lo standard Bluetooth opera nella banda libera ISM (Industrial,

Scientific and Medical) centrata attorno ai 2.4 GHz. Questa banda è occupata da un elevato numero di emettitori RF, dalle applicazioni wireless proprietarie, allo standard Wireless Ethernet (IEEE 802.11b), fino ad arrivare a generatori di rumore come i forni a microonde e le lampade al sodio da strada. Ciò ha reso necessario l'utilizzo di una tecnica di modulazione robusta alle interferenze: si è scelto una 2-FSK con il frequency hopping come tecnica di spread spectrum (FHSS). Questo sistema di comunicazione funziona secondo uno schema Time Division Multiplexing (TDM), dove l'unità base di operazione è uno slot di durata pari a 625 μs. Quando i dispositivi sono connessi, negli slot pari è abilitato a trasmettere il master, mentre negli slot dispari sono abilitati a rispondere gli slave, uno alla volta; in particolare può rispondere lo slave che aveva ricevuto dati dal master nello slot precedente. L'utilizzo della tecnica FHSS comporta che ogni time slot sia caratterizzato da una frequenza di trasmissione differente, secondo una sequenza di valori frequenziali decisi dal master seguendo un particolare algoritmo di calcolo del frequency hopping. Il bitrate massimo raggiungibile in aria è pari a 1 Mbps.

Classi di potenza

Le specifiche Bluetooth definiscono tre classi di potenza e le relative distanze che si possono raggiungere in trasmissione. Ciò è mostrato nella Tabella 1.1:

Classe potenza	**Potenza output (max)**	**Distanza (max)**
1	100 mW (20 dBm)	100 m
2	2.5 mW (4 dBm)	10 m
3	1 mW (0 dBm)	1 m

Tabella 1.1: Classi di potenza

I dispositivi di Classe 1 hanno d'obbligo il requisito del controllo di potenza, che è opzionale per le classi 2 e 3. Comunque, per un minimo consumo di potenza, è preferibile avere il controllo di potenza.
Il controllo di potenza agisce tramite un ricevitore che esamina un segnale di monitorazione della potenza, l'RSSI (Receiver Signal Strength Indication); in caso di segnale troppo debole viene mandato un segnale di controllo LMP_incr_pow (a livello Link Manager), che richiede di aumentare la potenza per avere un canale efficiente, mentre in caso di segnale più forte del necessario, c'è una richiesta di diminuire la potenza, con un segnale di controllo LMP_decr_pow_req.

Parametri di prestazione del sistema RF

Le specifiche Bluetooth forniscono dei parametri prestazionali per il sistema RF. Alcuni di questi parametri sono accettabili solo sulla carta, per cui molti dispositivi Bluetooth in commercio eccedono significativamente rispetto alle prestazioni di specifica.

Bluetooth deve operare con un Bit Error Rate (BER) massimo dello 0.1%. Questo significa avere un ricevitore con una sensibilità di -70 dBm, anche se nella realtà i ricevitori Bluetooth superano questa specifica di 10 dBm o più.

Le specifiche non forniscono valori per il tempo di assestamento nella sintetizzazione. Il processore del livello più basso prima decide in che stato mettere il controllore Baseband, che poi deve calcolare la nuova frequenza di trasmissione secondo la sequenza del FHSS e programmare il sintetizzatore. Questo impone un tempo di sintetizzazione di circa 180 μs, ma alcuni dispositivi in commercio hanno tempi migliori, compresi tra i 130 e i 170 μs.

Connessione di dispositivi Bluetooth

Per comunicare tra loro due device devono stabilire una connessione. Se si conosce l'indirizzo Bluetooth dell'altro dispositivo e l'identificativo del servizio, è possibile inoltrare una richiesta di connessione. I dispositivi si connetteranno se la loro distanza non è superiore alla massima consentita dal modulo installato e se i parametri di sicurezza lo consentono; questo implica chiaramente che il dispositivo a cui ci si connette deve possedere il servizio richiesto.

Inquiry

La prima cosa che deve fare un'applicazione di rete Bluetooth è ricercare gli altri dispositivi abilitati presenti nel proprio range di copertura dell'antenna. Per dispositivi abilitati si intendono tutti quei dispositivi Bluetooth che sono in modalità discoverable ossia in inquiry scan.

Il dispositivo che inizia questa procedura spedisce in modalità broadcast un messaggio di inquiry e attende che i dispositivi presenti nella zona rispondano. Per stabilire quando un dispositivo deve rispondere sono state definite tre modalità di discoverable (general, limited e not discoverable) e due diversi tipi di inquiry (general e limited).
Quando un dispositivo esegue una general inquiry tutti i dispositivi in modalità general e limited discoverable devono rispondere al messaggio. Nel caso in cui invece si esegua una limited inquiry solo i dispositivi in modalità limited discoverable risponderanno. I dispositivi che sono in modalità not discoverable non rispondono in nessun caso a messaggi di inquiry ed è come se fossero invisibili agli altri.

Per permettere di settare la modalità di discoverable le API Java JSR-82 [2] definiscono il metodo setDiscoverable() della classe LocalDevice.

L'argomento passato a setDiscoverable() rappresenta la modalità di discoverable che il dispositivo dovrà usare per rispondere a messaggi di tipo inquiry.
Per eseguire la ricerca vera e propria dei dispositivi si utilizzano i metodi della classe DiscoveryAgent che come vedremo si occupa anche della ricerca dei servizi. A tale scopo sono forniti due metodi: startInquiry() e retrieveDevices().
Come si intuisce facilmente il metodo startInquiry() da inizio al procedimento di ricerca e prende come argomento due parametri che rappresentano il tipo di inquiry da eseguire (general o limited) ed un DiscoveryListener. L'applicazione deve infatti creare una classe che implementi l'interfaccia DiscoveryListener e implementarne i metodi, che nel caso di inquiry sono deviceDiscovered() e inquiryCompleted().

Il metodo startInquiry() è una chiamata non bloccante e questo significa che ritorna prima del termine della procedura. E' per questo motivo che si rende necessario il DiscoveryListener; al termine dell'inquiry infatti l'implementazione delle JABWT (Java APIs for Bluetooth Wireless Technology) prevede che venga richiamato il metodo inquiryCompleted() della classe DiscoveryListener per mettere a conoscenza l'applicazione della conclusione della ricerca.

Il metodo deviceDiscovered() viene invece richiamato automaticamente ogni volta che viene scoperto un nuovo dispositivo e si riceve la sua risposta. A questo metodo viene passato come argomento un'istanza della classe RemoteDevice che appunto rappresenta il dispositivo remoto appena conosciuto e permette attraverso una serie di metodi di conoscerne alcune informazioni come l'indirizzo Bluetooth o il nome alfanumerico che lo identifica (user-friendly name).

L'altro metodo messo a disposizione dalla classe DiscoveryAgent per l'inquiry è retrieveDevices() che in realtà non esegue un'inquiry vera e propria ma si limita a ritornare quei dispositivi che erano già stati scoperti da inquiry precedenti e quindi non fornisce nessun tipo di garanzia sulla loro reale presenza. Con questo metodo è possibile anche inserire nella lista dei device ritornati, una serie di dispositivi di default che possono tornare utili nel caso in cui la topologia della rete è piuttosto stabile e non soggetta a grossi cambiamenti.

Service Discovery

Una volta che si conoscono i dispositivi presenti nell'area di "operabilità", il passo successivo è quello di determinare quali servizi questi mettono a disposizione; per fare questo si utilizza il Service Discovery Protocol definito dalle specifiche Bluetooth. Un'applicazione che vuole fornire dei servizi, e che per questo funge da server, deve permettere agli altri dispositivi di trovare questi servizi: li pubblica così tra i suoi service records i quali, attraverso una serie di attributi, li descrivono. Questi attributi specificano il nome del servizio, il modo con il quale connettersi ad esso, una breve descrizione del suo funzionamento ed altre informazioni utili al suo corretto utilizzo.

La procedura con la quale un server rende visibili i propri service records ai potenziali client prende il nome di service registration. Questa inizia con la chiamata a Connector.open() che restituisce uno StreamConnectionNotifier e crea un apposito service record per il nuovo servizio, con una serie di attributi di default già settati in base ad una stringa che viene passata come argomento al metodo Connetor.open() e che verrà descritta in dettaglio nel paragrafo successivo. Per ora basta sapere che in questa stringa è

contenuto anche l'identificatore globalmente univoco del servizi (UUID Universally Unique Identifier). Una volta creato il service record è possibile aggiungerci dei propri attributi con il metodo setAttributeValue() della classe ServiceRecord che prende come argomenti un identificatore e il valore dell'attributo sotto forma di istanza della classe DataElement.
Arrivati a questo punto il servizio non è ancora visibile agli altri dispositivi; per renderlo visibile occorre andare a modificare il contenuto del SDDB usando il metodo acceptAndOpen() sull'istanza della classe StreamConnectionNotifier ritornata da Connector.open(). Ora il servizio è correttamente registrato e il server è in attesa di client che si connettano.

Vediamo ora quali sono le operazioni, definite dal Service Discovery Application Profile, che devono compiere i client per trovare e connettersi ai servizi. Supponiamo che sia già stata eseguita la procedura di inquiry e che abbia ritornato come risultato alcuni dispositivi. Si vuole ora verificare se qualcuno di questi dispone di un particolare servizio al quale si è interessati e del quale già si conosce il suo UUID.
Per fare questo si deve richiamare su ognuno dei dispositivi trovati il metodo searchServices() della classe DiscoveryAgent che prende come argomento una lista degli UUID che si stanno cercando e una lista dei service attributes che devono essere ritornati per ogni service record trovato e identificato con uno degli UUID ricercati. Come nel caso della ricerca dei dispositivi anche in questo caso occorre implementare l'interfaccia DiscoveryListener (passando il Listener alla funzione searchServices()) e definirne i metodi servicesDiscovered() e serviceSearchCompleted().
La chiamata a searchServices() come startInquiry() non è bloccante e l'implementazione delle JSR-82 si preoccupa di richiamare il metodo servicesDiscovered() ogni volta che si ricevono dei service records di risposta.

Generalmente quando riceve la prima risposta, il client sospende la ricerca invocando il metodo cancelServiceSearch(). Nulla impedisce comunque di portare a termine la ricerca e anzi può essere utile farlo perché, nel caso in cui il servizio non sia più raggiungibile o funzionante, si potrebbero già avere delle altre possibili fonti sulle quali provare. In ogni caso al termine della ricerca viene automaticamente richiamato il metodo serviceSearchCompleted() che riporta lo stato d'uscita dell'operazione (ricerca completata correttamente, ricerca terminata dall'utente, errori, ecc.).
Se la ricerca è andata a buon fine il client è ora in grado, con le informazioni contenute nel/i service record/s, di creare delle connessioni con altri dispositivi che mettono a disposizione il servizio ricercato.

Comunicare con RFCOMM

Il protocollo RFCOMM fornisce l'emulazione di una linea seriale di tipo RS-232 tra due dispositivi Bluetooth. JSR-82 non definisce nessuna nuova classe o interfaccia per utilizzare RFCOMM ma si limita a riutilizzare delle classi e delle interfacce già esistenti e che fanno parte del GCF (Generic Connection Framework). Questa scelta permette di avere un'estrema flessibilità dal punto di vista dei protocolli di comunicazione utilizzati, perché come si vedrà permette di utilizzarne indifferentemente diversi senza quasi modificare il codice, e ne agevola il porting su piattaforme diverse.

Tutti i tipi di connessione iniziano con Connector.open() che riceve come parametro una stringa che identifica il protocollo da utilizzare e il dispositivo al quale connettersi. Questa stringa segue lo standard definito dal GCF per il formato delle stringhe di connessione secondo il quale la stringa è divisa in tre parti separate nel modo seguente:

schema://target;parametri

Lo schema rappresenta il protocollo da utilizzare e nel caso di RFCOMM è *btspp* dove *bt* sta per Bluetooth e *spp* per Serial Port Profile che è il profilo che realizza RFCOMM. Il target nel caso del client è l'indirizzo Bluetooth e l'identificatore del canale utilizzato dal service separati da ":", mentre nella stringa di connessione del server il target è la parola chiave *localhost* separata dall'Universally Unique IDentifier (UUID) del servizio anche in questo caso da ":". Infine la stringa si conclude con una serie di attributi opzionali tra i quali sono anche inclusi quelli relativi alle impostazioni di sicurezza.
Un esempio di stringhe di connessione RFCOMM valide per il client sono:
btspp://00803d000001:1
btspp://008034AD2AA1:3;authenticate=true

mentre per il server:

btspp://localhost:EFAA5621975548568F27B437B6C5E6B2
btspp://localhost:93007CA747114F42B1DCCE878A6531FA;name=MyService
Di queste URL, mentre quelle lato server possono venire generate anche staticamente a livello di implementazione (basta conoscere solamente l'UUID del service ed impostare i parametri desiderati), quelle lato client vengono costruite dinamicamente in quanto bisogna conoscere le informazioni relative al server al quale ci si vuole connettere. Queste informazioni vengono recuperate dai service records che si sono ottenuti durante la ricerca dei servizi disponibili sui dispositivi trovati in fase di inquiry richiamando il metodo getConnectionURL() dell'interfaccia ServiceRecord che ritorna la stringa di connessione completa per connettersi a quel servizio.

Quando la stringa di connessione lato client viene passata a Connector.open() questo ritorna un oggetto StreamConnection con il quale l'applicazione è in grado di crearsi gli stream di input ed output con

i rispettivi metodi openInputStream() ed openOutputStream() per poter comunicare con il server.

Se invece a Connector.open() viene passata una stringa di connessione del server (riconoscibile perché contiene la keyword localhost) allora viene ritornato un oggetto di tipo StreamConnectionNotifier. Utilizzando il metodo bloccante acceptAndOpen() di questa classe l'applicazione si mette in attesa di connessioni da parte dei client. Solo quando qualche client si connette, il metodo risveglia il processo restituendo un oggetto StreamConnection. Come nel caso del client, anche il server utilizza questo oggetto per ottenere gli stream di input e output per leggere e scrivere dati.

In questo paragrafo abbiamo visto come si stabiliscono delle connessioni utilizzando RFCOMM. In realtà la scelta di riutilizzare le librerie del GCF rende questa procedura standardizzata a diversi protocolli di comunicazione. Per esempio nel caso in cui si volesse sostituire il protocollo RFCOMM con L2CAP l'unica cosa che dovrebbe essere modificata è la stringa di connessione che viene passata a Connector.open(), nella quale bisognerebbe semplicemente sostituire lo schema *btspp* usato per RFCOMM con *btl2cap* usato invece da L2CAP.

Capitolo 2

ToothAgent: Architettura di sistema

Al fine di sopperire alla mancanza di tool basati su piattaforme multi-agente in cui gioca un ruolo essenziale sia la collaborazione che la competizione degli agenti, abbiamo sviluppato *ToothAgent*, un sistema in grado di supportare un numero pressoché infinito di servizi; non si limita quindi a fornire, come invece fanno i tool esistenti, semplici indicazioni o informazioni turistiche, ma al contrario, supportando la collaborazione fra agenti, diviene estendibile ed utilizzabile per qualsiasi servizio. In tal modo esso si colloca in una posizione privilegiata rispetto ai progetti presentati nella sezione 1.2 e introduce nuovi scenari collaborativi.

In questo capitolo, dopo una descrizione generale del sistema *ToothAgent* e dei suoi requisiti funzionali e non, verrà presentata l'architettura generale: si illustreranno quindi le varie sotto-componenti entrando nel dettaglio delle loro interazioni e interconnessioni. Si suggeriranno inoltre alcuni possibili scenari d'utilizzo (mostrando come il sistema possa essere replicato

in un qualsiasi altro ambiente e con diverse tipologie di utilizzatori) e si presenteranno i servizi dimostrativi implementi. Prima di concludere parlando delle modalità di recupero delle risposte pendenti e delle piattaforme ad agenti, si presenterà il framework di Cultura Implicita e si mostreranno i benefici derivanti dalla sua introduzione nel sistema.

2.1 L'idea del sistema *ToothAgent*

L'idea di *ToothAgent* nasce dall'assenza di applicazioni capaci di permettere la collaborazione e la competizione di agenti, e dalla volontà di costituire una comunità virtuale attiva e dinamica utile alle persone. I concetti base del progetto sono quelli di mobilità, ubiquità, interazione, ma soprattutto località. Ognuna di queste nozioni verrà specificata meglio in seguito. Per ora ci limitiamo a dire che *ToothAgent* è un sistema per la comunicazione, coordinazione, collaborazione e competizione fra utenti; utenti che però, pur essendo persone reali, giocano nel sistema un ruolo prettamente virtuale e sono rappresentati da agenti che si comportano, nelle varie situazioni, come si comporterebbero loro.

Come prima cosa si è studiata un'architettura indipendente dai servizi offerti in grado di supportare gli utenti; saranno i servizi, ovvero i temi su cui sarà focalizzata l'interazione, che poi costruiranno attorno agli utenti una comunità nella quale sia possibile il confronto e il contatto (non diretto ma virtuale) con altre persone per una costante informazione.

Le diverse tipologie di servizio offerte da *ToothAgent* creeranno svariate sotto-comunità autonome, ognuna incentrata su un tema differente e legata ad uno specifico argomento di discussione; l'implementazione dei servizi, sarà libera e dovrà rispettare solo pochi vincoli per essere conforme al sistema

stesso. Ognuno di questi servizi richiederà agli utenti una configurazione personale che verrà fatta in base alle loro preferenze ed ai loro desideri.

Si immagini di spargere sul territorio una serie di server, tutti collegati fra loro via internet. Ognuno di questi server supporta uno o più servizi inerenti il luogo in cui sono posti e la tipologia di persone che frequenta quel luogo, e può comunicare con altri dispositivi mobili (cellulari e PDA) attraverso una connessione wireless Bluetooth. Il raggio di azione è dunque ristretto e la comunicazione può avvenire solo se il dispositivo mobile (e quindi la persona) si trova nelle immediate vicinanze del server.

Ogni utente, interfacciandosi al sistema con il proprio dispositivo, può inviare ai server una sorta di richiesta, dall'interpretazione della quale vengono creati degli agenti che lavoreranno in modo attivo ed autonomo sulla piattaforma multi-agente installata sul server. Il lavoro di questi agenti prevede il dialogo con altri agenti presenti sulla piattaforma: da questo dialogo l'agente ricava alcune informazioni utili al proprio utente; informazioni che sono ovviamente ricavate sulla base delle preferenze, indicate nella richiesta fatta in precedenza. Esse vengono quindi trasferite dall'agente al dispositivo mobile dell'utente che potrà finalmente consultarle ed utilizzarle come meglio ritiene.

Gli agenti, a seconda della tipologia di servizio, non si limitano a dialogare con gli altri agenti presenti sulla piattaforma, ma devono in alcuni casi competere per il raggiungimento di alcuni obiettivi; altre volte possono invece collaborare e negoziare.

L'utente non potrà per forza di cose essere collegato al sistema sempre ed ovunque (a causa del carattere localizzato di quest'ultimo), ma dovrà avere la garanzia di interfacciarsi ad esso in modo rapido ed automatico ogni volta si trovi nel raggio di azione di un punto di accesso del sistema. Sfruttando il

fatto che i propri agenti continuano a vivere e lavorare nonostante l'utente sia uscito dal raggio d'azione del server su cui essi operano, e grazie alla connessione dei server via internet, l'utente potrà in ogni caso recuperare le informazioni per lui elaborate in qualsiasi momento; gli basterà interfacciarsi ad un qualunque server facente parte del sistema.

In definitiva si può pensare lo scenario descritto come una sola grande comunità di utenti, nella quale esistono dei sottogruppi (identificabili con i diversi servizi offerti) che trattano e che sono interessati ad argomenti differenti. Ogni sottogruppo avrà una propria autonomia, ma dovrà collaborare con gli altri gruppi al fine di scambiare messaggi ed informazioni utili al buon funzionamento del sistema.

2.2 Requisiti del sistema

Il sistema deve soddisfare un numero di requisiti che possono essere riassunti come di seguito:

- Il sistema dovrà essere aperto a tutti e sarà il meno invasivo possibile; dovrà perciò essere semplice da utilizzare e non dovrà richiedere particolare dimestichezza o esperienza con i dispositivi mobili. Ogni azione e scambio di messaggi dovranno essere completamente automatizzati e all'utente non dovranno essere richieste particolari abilità per la configurazione e l'attivazione dei servizi.

- Saranno fornite due applicazioni: una che sarà installata sul PC dell'utente e l'altra che verrà invece installata sul dispositivo mobile. Vi sarà poi un sito internet attraverso cui l'utente dovrà registrarsi fornendo i propri dati personali e quelli del dispositivo mobile che utilizzerà. Dal sito si potrà scaricare, oltre alle applicazioni, anche un

file contenente le informazioni di tutti i server che offrono dei servizi: tale file sarà quello usato per la configurazione dei servizi stessi.

- Il sistema dovrà permettere agli utenti di esprimere i loro interessi e di scegliere i servizi a cui essi vogliono accedere. Ciò significa che un utente dovrà avere la possibilità di ricercare i servizi a cui è interessato in base ad alcune loro caratteristiche; la ricerca potrà quindi essere compiuta facendo riferimento a tutti i server che sono installati in una particolare città, o in un particolare edificio; potrà essere effettuata filtrando i servizi in base alla categoria del servizio oppure in base ad una keyword contenuta nella descrizione del servizio stesso. Tale "scrematura" deve poter essere fatta tanto sul sito web (nella pagina dedicata al download e quindi prima di scaricare il file di configurazione) quanto utilizzando il programma installato sul PC. Questo significa che l'utente potrà quindi scegliere di scaricare l'intero file di configurazione (contenente tutti i server facenti parte del sistema) oppure un file ridotto (contenente solo un sottoinsieme dei server). I servizi configurati saranno visibili sull'interfaccia grafica dell'applicazione per il PC e saranno modificabili e cancellabili. Inoltre, tale configurazione dovrà poter essere facilmente trasferibile sul dispositivo mobile dell'utente, senza che egli debba in seguito configurare nulla.

- Il sistema dovrà garantire l'accesso ai servizi richiesti quando il dispositivo mobile si trova nelle immediate vicinanze del server. Ovviamente tale distanza dipenderà molto dalle caratteristiche del telefono cellulare o del computer palmare dell'utente, ma lo scambio di informazioni fra i dispositivi dovrà essere automatico e istantaneo. Il

sistema dovrà quindi essere in grado di supportare la ricerca continua di server che offrono uno o più determinati servizi e dovrà garantire che le preferenze dell'utente (contenute nel file di configurazione scaricato sul dispositivo mobile) vengano trasferite sul server con cui il dispositivo sta comunicando, solo se esso supporta uno o più servizi scelti dall'utente. Il file di configurazione dell'utente non dovrà quindi essere passato per intero al server, ma dovranno essere passati ad esso solamente i parametri che lo interessano. Per contro il server, ricevuto l'indirizzo Bluetooth del dispositivo e la password scelta dall'utente dovrà verificare, prima di accettare qualsiasi altra informazione, la validità dei dati ottenuti, ovvero verificare che il dispositivo mobile sia correttamente registrato e che la password corrisponda a quella inserita in fase di registrazione.

- Le richieste dell'utente dovranno essere trasferite al server in maniera automatica, così come in maniera automatica dovrà avvenire la creazione dell'agente. Per ogni utente dovrà essere creato un solo agente per ogni server; ciò vuol dire che se l'utente desidera più servizi che risiedono sullo stesso server, dovrà esserci un solo agente e questo si dovrà occupare contemporaneamente della gestione di più servizi. Il personal agent dell'utente si occuperà dell'interazione con gli altri agenti e della memorizzazione delle informazioni sul database del server su cui sta lavorando.

- Il sistema dovrà permettere all'utente di scaricare la proprie risposte pendenti (risposte ottenute e memorizzate sui database, ma non ancora scaricate) sia che esso sia ancora nel range d'azione del server su cui ha attivato il proprio agente, sia che ne sia uscito. Gli dovrà quindi

essere data la possibilità di accedere al database di qualsiasi server visitato per mezzo di un qualsiasi altro server facente parte della rete dei server che offrono dei servizi. Contattando via Bluetooth il server che si trova nelle immediate vicinanze, l'utente potrà recuperare le informazioni presenti sugli altri server; tutto questo avverrà ancora una volta in maniera completamente automatica.

Il sistema dovrà inoltre permettere all'utente di scaricare le risposte pendenti anche dal proprio PC utilizzando l'interfaccia dedicata. Per fare questo l'utente dovrà dapprima scaricare la lista dei server visitati sul proprio PC; l'applicazione, in seguito, connettendosi via internet ai database dei vari server, recupererà i dati richiesti e li memorizzerà in un apposito file. Come in precedenza, l'operazione avverrà in modo automatico, senza interazione da parte dell'utente, che si limiterà ad abilitare la comunicazione fra PC e dispositivo mobile.

- L'utente dovrà poter avere la possibilità di verificare che il proprio file di configurazione sia aggiornato ed eventualmente scaricare sul proprio PC il nuovo file.

- Il sistema dovrà garantire all'utente la possibilità di modificare i propri dati personali forniti in fase di registrazione. In qualsiasi momento quindi l'utente potrà per esempio modificare il proprio numero di telefono o aggiungere un nuovo dispositivo mobile a quelli già esistenti.

- Il sistema dovrà prevedere un livello di sicurezza adeguato, garantendo all'utente che lo utilizza la riservatezza dei propri dati e delle risposte ottenute con l'utilizzo dei vari servizi. Per permettere questo, dovrà essere richiesta una registrazione nella quale, oltre ai propri dati personali, devono essere forniti anche l'indirizzo Bluetooth

dell'apparecchio con cui si intende accedere ai servizi ed una password, che dovrà essere utilizzata ogni qualvolta l'utente voglia accedere ad un qualche database per il recupero delle proprie risposte o ogni qualvolta voglia ricercare qualche server per attivare i propri servizi. Ciò significa che la password dovrà essere inserita in un'apposita schermata presente nel programma dedicato al cellulare/palmare all'inizio di ogni nuova sessione. I dati personali forniti in fase di registrazione potranno essere utilizzati dai vari servizi per reperire informazioni riguardanti l'utente, come per esempio il suo nome, la sua età, il suo recapito telefonico o la sua e-mail.

Al software, sebbene assolutamente non invasivo in quanto non dovrà richiedere nessuna particolare esperienza o dimestichezza con i dispositivi mobili e necessiterà solo di una minima interazione da parte dell'utente , potrebbe essere sollevata un'obiezione: l'utente potrebbe infatti trovarsi in difficoltà se posto di fronte a delle scelte da prendere per configurare un determinato servizio in maniera corretta, o meglio, in maniera ottimale, e se non lo facesse, potrebbe non ottenere alcun risultato. Per ovviare a questo problema si è pensato di introdurre un ulteriore requisito:

- Le piattaforme dovranno permette ai nuovi membri di una comunità di comportarsi in accordo con la cultura (ovvero con le usanze) della comunità stessa. Non conoscendo i parametri ottimali da inserire, l'utente potrà quindi delegare al software il compito di completare le proprie preferenze, in base a delle politiche variabili da servizio a servizio, ma che si baseranno sempre sul comportamento degli altri utenti che popolano la piattaforma. Le piattaforme dovranno quindi essere implementate basandosi sul framework di Cultura Implicita [15] che bene si addice alla risoluzione di questo tipo di problemi.

L'utente dovrà comunque avere in ogni caso la possibilità di scelta, decidendo se avvalersi o meno dell'aiuto della Cultura Implicita per la configurazione dei propri servizi.

2.3 Componenti architetturali

L'architettura del sistema include quattro componenti principali: i dispositivi mobili, i PC, i server e i database. In questa sezione daremo una descrizione di ogni singola componente, indicando le principali caratteristiche.

La Figura 2.1 illustra l'architettura generale e le interazioni tra i componenti. La connessione fra i dispositivi mobili e i PC e fra i dispositivi mobili e i server viene stabilita via Bluetooth, mentre quella fra PC e server e quella fra server e server vengono stabilite utilizzando la rete Internet, utilizzando una connessione via cavo oppure wireless.

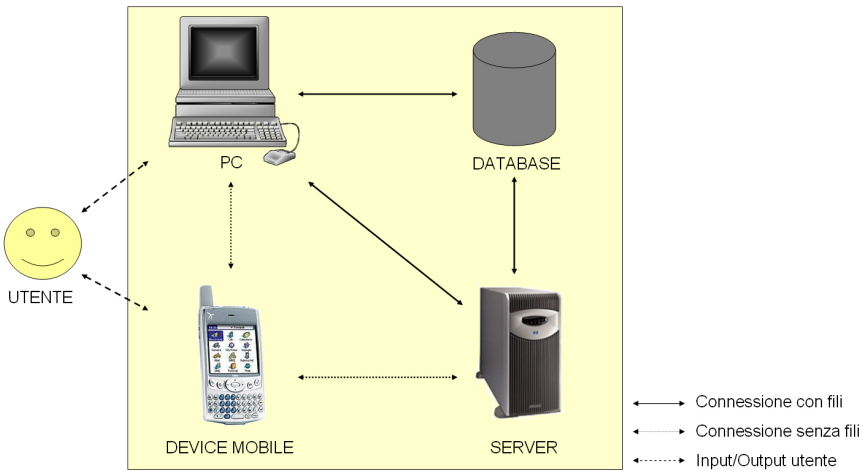


Figura 2.1: Interazione dei vari componenti del sistema

- La componente per il *PC* è un programma scritto in Java che gestisce la configurazione dei servizi e permette lo scambio e il recupero delle informazioni; l'applicazione fornisce un'interfaccia che gestisce la ricerca, la selezione, e l'impostazione dei servizi. Essa permette inoltre il recupero e la visualizzazione delle risposte pendenti e l'invio del file di configurazione ai dispositivi mobili.

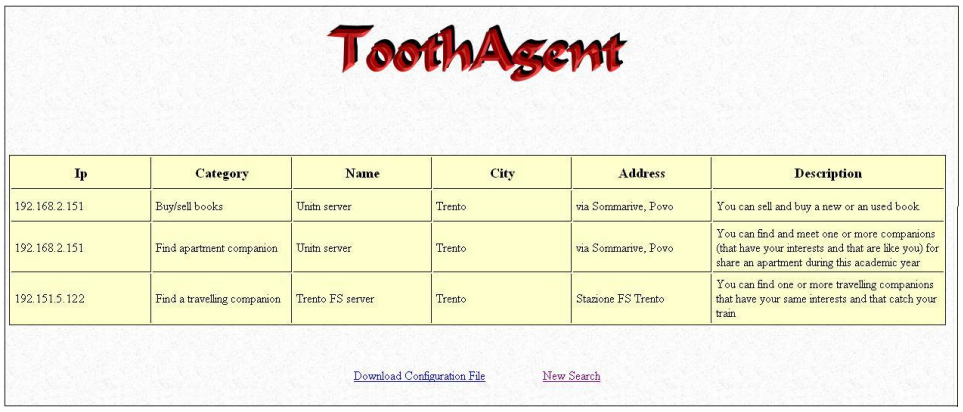
- I *dispositivi mobili* vengono utilizzati come ponte fra i PC e i server, ovvero per l'invio delle richieste dell'utente - necessarie per la creazione degli agenti, e per la ricezione dei dati di risposta. L'applicazione installata sui dispositivi mobili permette inoltre di visualizzare le risposte ricevute e i server visitati e di inviare tali dati al PC. Il menu è stato appositamente creato semplice ed intuitivo; tuttavia vengono offerte numerose schermate per la visualizzazione, cancellazione e impostazione dei servizi, delle risposte e dei server visitati. L'utente è sempre in grado di osservare lo stato della connessione - sia che il dispositivo stia cercando nuovi server per lanciare i propri agenti, sia che stia cercando risposte pendenti - e di attivarla o interromperla in qualsiasi momento.

- Ogni *server* facente parte del sistema fornisce uno o più servizi; su ognuno di essi è inoltre installata una piattaforma multi-agente - ospitante i personal agent che rappresentano i singoli utenti - e un *database* - per la memorizzazione degli agenti attivi e delle risposte ottenute. Grazie all'indipendenza del sistema creato, è possibile utilizzare una qualsiasi piattaforma multi-agente con un qualsiasi database su un qualsiasi sistema operativo, ignoti all'utente finale del programma. Egli infatti non dovrà preoccuparsi in nessun caso di

questi particolari e potrà continuare a utilizzare il sistema senza la necessità di configurare nulla. L'applicazione installata sul server si occupa di comprendere le richieste dell'utente e di ridirigere l'input ottenuto al rispettivo personal agent, aggiungendo ad esso il nuovo servizio (per ogni utente infatti viene creato un solo agente che si occuperà di tutti i servizi richiesti), o creandone uno nuovo.

- Il *database centrale*, accessibile esclusivamente dai server via web, tiene memoria sia delle informazioni degli utenti registrati, sia delle proprietà di tutti i server che offrono servizi (come il loro indirizzo IP, il loro nome, la loro locazione e i parametri che essi richiedono). Ogni qualvolta vengano aggiunti nuovi servizi, il database centrale viene aggiornato e viene creato un nuovo file di configurazione scaricabile dagli utenti, che così facendo possono aggiornare quello ormai obsoleto, ma ancora perfettamente funzionante, presente sui loro PC.
  Sul server centrale sono caricate inoltre delle pagine php alle quali si accede tramite internet e che servono per la registrazione al sistema. La registrazione dell'utente al sistema avviene infatti tramite l'accesso ad alcune pagine web che gestiscono la memorizzazione dei dati sul database centrale e permettono il download dei programmi che andranno installati rispettivamente sul PC e sul dispositivo mobile. Tali pagine permettono in aggiunta di eseguire ricerche di servizi in base ad alcuni parametri quali la locazione del server, la tipologia del servizio offerto, ecc. (Figura 2.2) e di visualizzare la lista completa dei server (Figura 2.3).

Figura 2.2: La ricerca dei servizi può essere filtrata attraverso quattro differenti parametri



Ip	Category	Name	City	Address	Description
192.168.2.151	Buy/sell books	Unitn server	Trento	via Sommarive, Povo	You can sell and buy a new or an used book
192.168.2.151	Find apartment companion	Unitn server	Trento	via Sommarive, Povo	You can find and meet one or more companions (that have your interests and that are like you) for share an apartment during this academic year
192.151.5.122	Find a travelling companion	Trento FS server	Trento	Stazione FS Trento	You can find one or more travelling companions that have your same interests and that catch your train

Figura 2.3: Esempio di server installati nella città di Trento

2.4 Scenari d'utilizzo e accorgimenti

Il sistema così com'è stato presentato può essere replicato e contestualizzato in diversi ambiti. Per esempio lo si può immaginare installato in ogni facoltà di un ateneo, nella biblioteca comunale, nei bar, oppure nelle stazioni ferroviarie o nelle agenzie di viaggio, e più in generale in qualsiasi ambiente in cui si voglia offrire agli utenti/clienti un particolare servizio; ampliando le ambientazioni viene formato uno scenario più esteso ed eterogeneo dal punto di vista dell'utilizzo che ne viene fatto.

Una delle componenti principali del sistema è costituita dai device mobili. Un sistema comprendente questo tipo di dispositivi, deve sicuramente tener conto delle problematiche relative allo spostamento degli utenti da un luogo ad un altro e contemporaneamente del carattere al contempo locale e distribuito; locale perché i modi con i quali i cellulari ed i palmari possono comunicare attraverso l'uso del Bluetooth è limitato ad un raggio piuttosto ristretto (come descritto in 1.3 varia fino ad un massimo di 100 metri, in base al dispositivo); distribuito per rendere disponibili più punti d'accesso allo stesso sistema in modo da "seguire" l'utente nei suoi spostamenti nei punti in cui potrebbe tornare utile ai fini dell'applicazione.

Un possibile scenario di utilizzo dell'applicazione potrebbe essere una struttura universitaria nella quale vi sono utenti che, per il loro ruolo, possono essere interessati a svariati servizi inerenti i loro studi o i loro hobby. In Figura 2.4 è riportato un esempio di come i server potrebbero essere distribuiti proprio all'interno di una facoltà e di come potrebbero essere tra loro collegati.

Un altro scenario d'utilizzo potrebbe essere una stazione ferroviaria, che potrebbe mettere a disposizione dei passeggeri servizi di informazione sui treni o sulle città di destinazione, avvisi di ritardi e/o soppressioni, ma

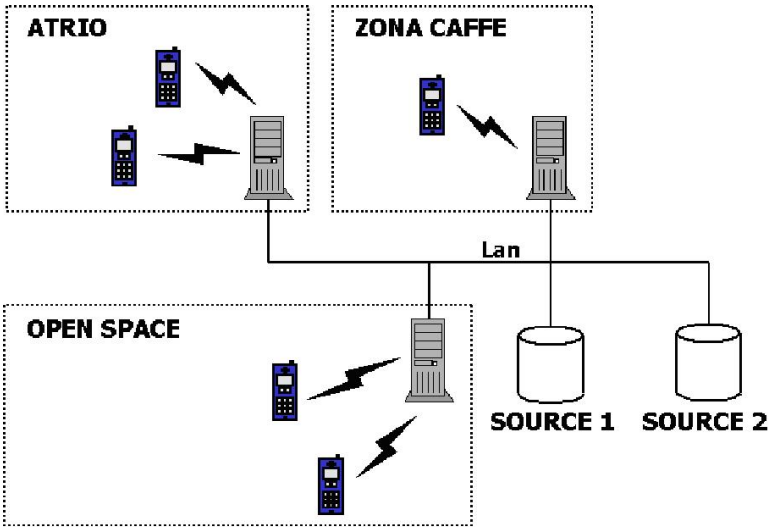


Figura 2.4: Esempio di distribuzione degli access-point in un edificio universitario

anche servizi di intrattenimento e svago.

Un'agenzia di viaggi potrebbe infine mettere a disposizione un servizio per l'organizzazione di viaggi di gruppo, o per segnalare agli interessati delle offerte speciali, che possono essere appositamente create per gli utilizzatori del servizio; o ancora potrebbe offrire degli sconti speciali per tutti quelli che, indipendentemente, si organizzano in gruppi per effettuare escursioni o gite avventurose.

Si può pensare di installare, in ognuno di questi luoghi, uno o più punti di comunicazione con il sistema in diversi posti all'interno dei vari edifici. Ognuno di questi access-point è in grado, via Bluetooth, di accettare richieste dagli utenti, elaborarle e fornire le risposte adeguate. Ovviamente per avere una situazione come quella appena descritta è necessario che gli access-point siano collegati tra loro attraverso una rete cablata o wireless che permetta di condividere le informazioni relative al sistema ed agli utenti. La Figura 2.5 mostra come le varie piattaforme facenti parte del sistema

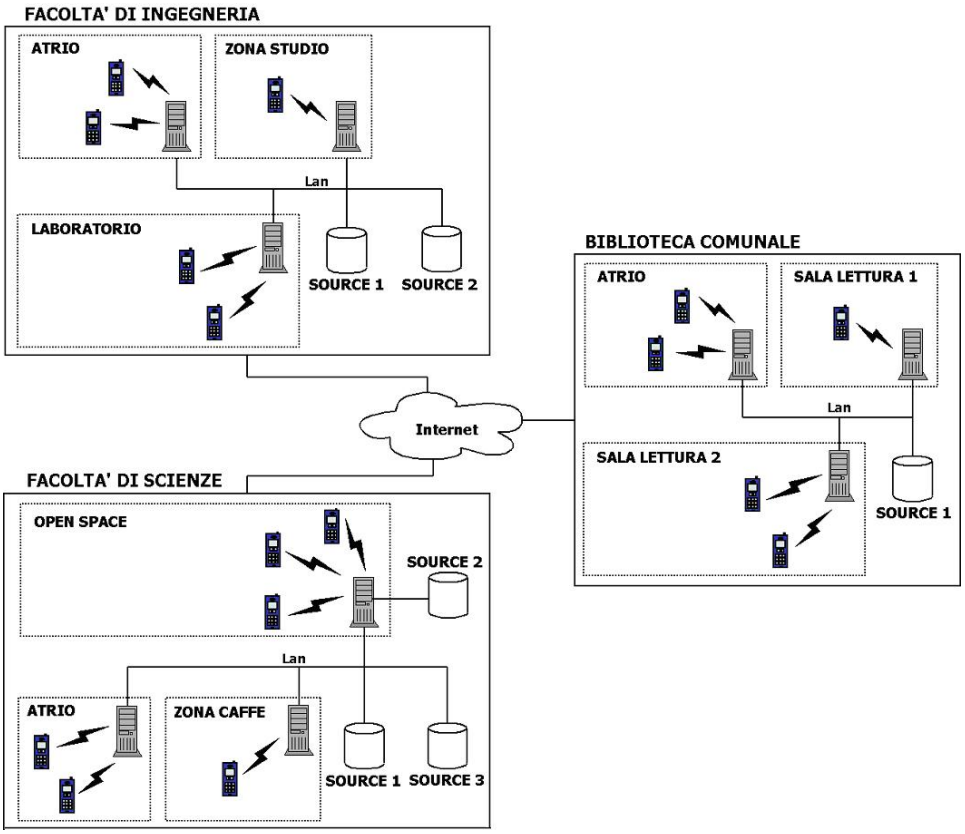


Figura 2.5: Replicazione del sistema in contesti diversi

possano essere collagate fra loro e come il sistema possa essere facilmente replicato in qualunque posto si voglia e con qualunque servizio si voglia mettere a disposizione degli utenti.

Come già detto inoltre, poiché gli utenti e i loro dispositivi sono mobili, ovvero si spostano da una zona all'altra in continuazione, molte volte uscendo dal raggio di azione degli access point, è necessario tenere traccia dei loro "passaggi" e dei loro accessi; questo compito è assegnato ai dispositivi mobili che, ogni qualvolta stabiliscono una connessione con qualche server, devono memorizzare il suo indirizzo IP. Tutto ciò permetterà all'utente di visualizzare in un secondo momento tutte le informazioni elaborate dai propri agenti sulle varie piattaforme. L'utente potrà inoltre venire a

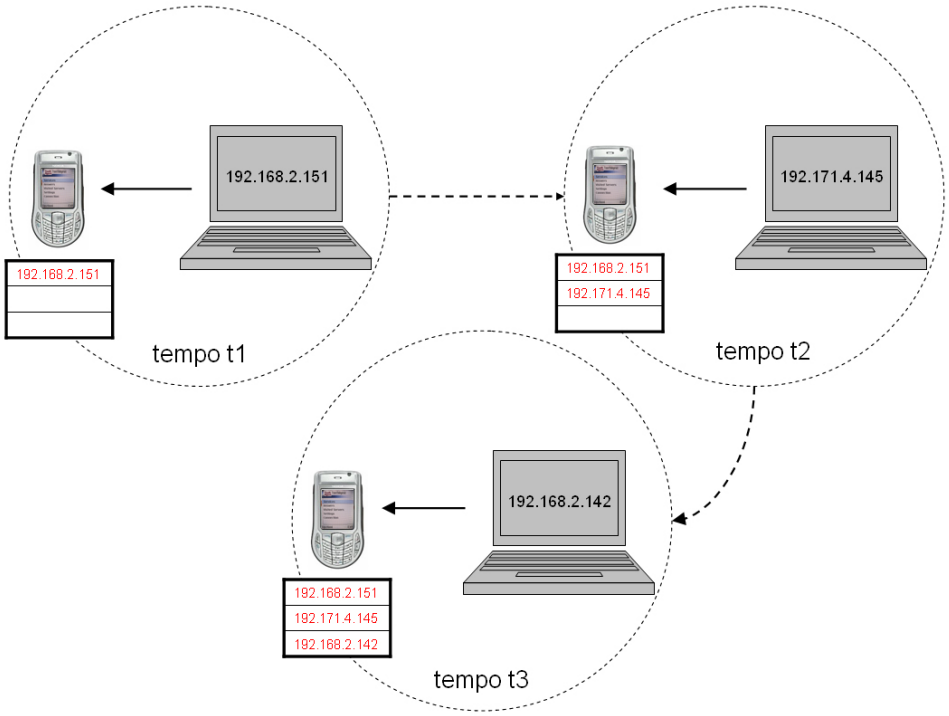


Figura 2.6: Ricezione e memorizzazione dell'indirizzo IP di tre diversi server in tre diversi momenti

conoscenza di altri server, di cui magari ignorava l'esistenza e tramite il loro indirizzo potrà risalire ai servizi che essi offrono: l'indirizzo IP del server viene infatti memorizzato ad ogni nuovo contatto, anche se il server non è fra quelli per cui l'utente ha settato i parametri per lanciare i propri agenti. La Figura 2.6 mostra un esempio di come avviene la memorizzazione degli indirizzi IP quando il dispositivo mobile dell'utente entra nel raggio d'azione di un server e lo contatta.

L'utente non può sempre pianificare la propria giornata e sapere esattamente dove si recherà e quali server visiterà. Il servizio scelto non deve quindi per forza di cose essere legato ad uno e un solo server ed essere offerto solamente da questo. Esso potrebbe essere replicato su n server, senza per questo il bisogno di dover configurare n servizi differenti in maniera uguale.

In questo caso, quando si ricercherà il servizio specifico, si otterrà una lista di più server e su ognuno di questi sarà possibile attivare i propri agenti senza la necessità di dover apportare modifiche alle configurazioni fatte in precedenza.

2.5 Accesso ai servizi

In questa sezione vengono descritti i processi per l'accesso ai servizi e viene data una rappresentazione di come essi avvengono e di come sono organizzati (Figura 2.7).

L'utente, prima di utilizzare il sistema vero e proprio, ovvero la ricerca dei server e il lancio dei propri agenti, effettua una ricerca dei servizi disponibili utilizzando l'interfaccia del programma installato sul PC (passo 1). La ricerca, come accennato in precedenza, può essere filtrata in base ad alcune preferenze dell'utente quali la locazione del server (ovvero la città in cui il server è installato), il tipo di servizio offerto oppure in base ad una parola chiave contenuta nella descrizione del servizio stesso. L'utente può così osservare a video i servizi disponibili ottenuti come risultato della ricerca compiuta (passi 2-3). Si passa così alla fase di configurazione dei servizi, settando i parametri richiesti e memorizzando il file che viene creato (passo 4). Il passo 5 corrisponde all'invio del file di configurazione al dispositivo mobile utilizzando la connessione wireless Bluetooth; una volta che il cellulare/PDA ha ricevuto e memorizzato il file, l'utente può dare inizio al vero e proprio servizio, ricercando i server desiderati e per i quali sono stati inseriti gli appositi parametri. Tale ricerca, automatica e continua, prosegue sino a quando non viene individuato un server. A questo punto viene stabilita una connessione fra il dispositivo mobile e il server (passo 6) e vengono inviati i dati necessari per la creazione e l'attivazione di un

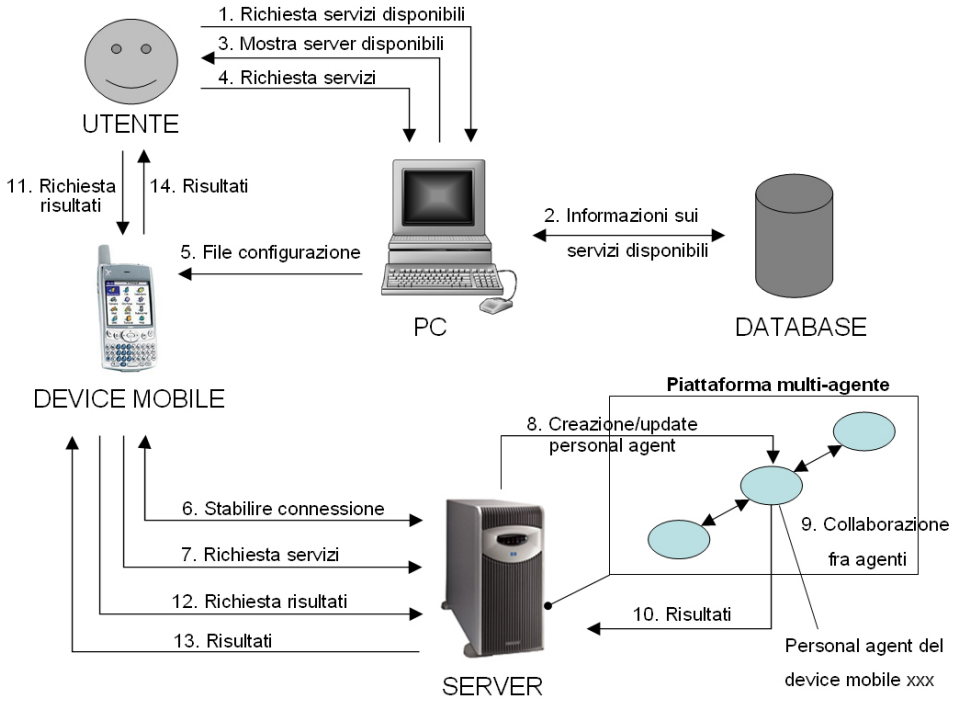


Figura 2.7: Sequenza temporale per l'accesso ai servizi

nuovo agente o per l'aggiornamento di un agente già esistente (passo 7). Il server, ottenuti i dati tramite la connessione wireless Bluetooth, verifica l'esistenza o meno di un agente che abbia come identificativo lo stesso indirizzo Bluetooth ricevuto e provvede alla creazione o all'aggiornamento del personal agent (passo 8); da questo momento ha inizio il periodo di vita del personal agent (che varia in base ai servizi che esso deve svolgere) il quale comincerà la ricerca di altri agenti con i quali collaborare, competere e/o contrattare (passo 9). Una volta individuati altri agenti e una volta conclusa la collaborazione/ competizione il personal agent può memorizzare i risultati ottenuti sul database del server (passo 10) e, a seconda del servizio, può terminare il proprio lavoro o proseguire nella ricerca di altri agenti. Quando l'utente desidera visualizzare le eventuali risposte ottenute, attiva sul proprio dispositivo mobile l'apposita ricerca (passo 11). Il cellulare/PDA

inizierà così la ricerca di un qualsiasi server facente parte del sistema e, una volta individuato, gli invierà la propria lista degli IP relativa ai server visitati (passo 12); il server, scorrendo progressivamente la lista, accederà via internet ai database dei server e recupererà le risposte pendenti dell'utente inviandole successivamente al suo dispositivo mobile (passo 13) che le memorizzerà e le renderà disponibili per la visualizzazione (passo 14). Un altro metodo per il recupero delle informazioni (non mostrato in figura ma dettagliatamente descritto in seguito nella sezione 2.6) consiste nell'accedere ai database direttamente dal PC tramite rete Internet: l'utente in quel caso deve prima trasmettere la lista degli IP visitati al PC (tramite Bluetooth) e successivamente, dopo avere inserito i propri dati personali per le opportune verifiche, inviare la richiesta. Tutte le informazioni ricevue saranno in questo caso salvate sul PC e saranno visualizzabili dall'interfaccia grafica del programma.

2.6 Scaricare i risultati pendenti

In questa sezione verranno descritte le due alternative per il recupero delle informazioni pendenti, ovvero delle risposte ottenute e memorizzate dagli agenti, ma non ancora scaricate e visualizzate dall'utente.

Si osservi la Figura 2.8 che riporta nel dettaglio i vari passi necessari per il recupero delle risposte memorizzate sui database.

L'utente ha sostanzialmente due opzioni per ricevere i risultati delle sue richieste. Il primo è quello di ricevere le informazioni direttamente sul proprio dispositivo mobile (passi A.xx). Tuttavia questa non è l'unica possibilità: l'utente può infatti abbandonare l'area coperta dal Bluetooth e non entrare in nessun altro raggio di copertura di un access point, oppure può non avere memoria sufficiente sul proprio cellulare/PDA per memorizzare le

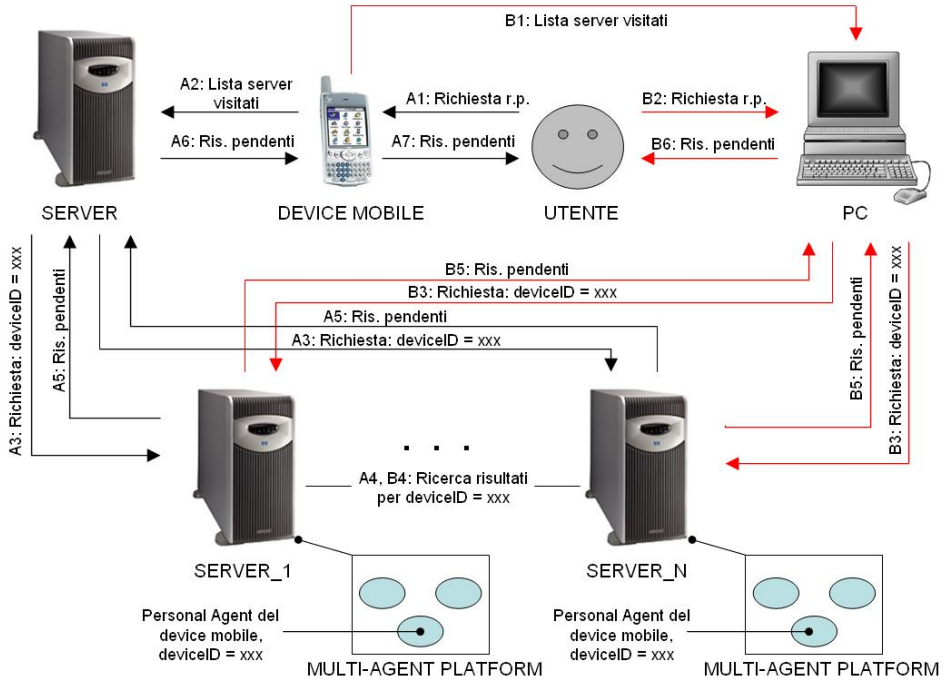


Figura 2.8: Il recupero dei dati pendenti utilizzando il dispositivo mobile (passi A.xx) e il PC (passi B.xx)

risposte (per esempio nel caso di un file grande o di più file). La seconda opzione è quindi quella di ricevere le informazioni direttamente sul PC (passi B.xx).

Nel caso l'utente voglia ricevere le risposte direttamente sul proprio dispositivo mobile, deve avviare il servizio di ricerca nella relativa schermata dell'applicazione presente sul cellulare/PDA. Attivando questa opzione (passo A1), inizia una continua e ciclica ricerca di server facenti parte del sistema. Una volta individuato un server, il dispositivo mobile invia, utilizzando la connessione Bluetooth, la lista dei server visitati dall'utente (passo A2); il server, scorrendo da lista degli IP, contatta uno dopo l'altro i server facendo richiesta dell'invio dei risultati pendenti dell'utente (passi A3). Ogni server contattato verifica la presenza delle risposte sul proprio

database (passi A4) e le restituisce al server che li ha contattati (passi A5). I risultati vengono inviati, ancora una volta utilizzando la connessione Bluetooth, al dispositivo mobile dell'utente (passo A6) e qui vengono memorizzati e resi disponibili per la visualizzazione (passo A7).

Le operazioni per il recupero delle risposte dal PC è simile ma, poiché la comunicazione non avviene più tra il dispositivo mobile e il server, ma tra il PC e i server, la lista degli IP relativa ai server visitati dall'utente deve essere scaricata sul computer. Tale operazione (passo B1) viene effettuata attivando la connessione Bluetooth tanto sul cellulare/PDA quanto sul PC. Prima di dare il via al download delle risposte dai server viene inoltre richiesto all'utente l'inserimento dei propri dati personali (login, ovvero indirizzo Bluetooth del dispositivo, e password) che serviranno per la verifica dell'effettiva registrazione al sistema e per il recupero delle corrette risposte dai server (tale operazione è visibile graficamente in Figura 3.9 a pagina 73).

In entrambi i casi - recupero delle risposte pendenti dal cellulare o dal PC - l'utente ha sempre la possibilità di visualizzare le risposte ottenute sullo schermo del computer; mentre però nel secondo caso la visualizzazione è immediata, poiché le risposte arrivano direttamente dai server contattati e sono visibili nell'interfaccia grafica del programma, nel caso l'utente abbia ricevuto le risposte sul proprio dispositivo mobile, dovrà trasferirle sul PC utilizzando la relativa funzionalità messa a disposizione dall'applicazione installata sul device e, ancora una volta, attivare la ricezione Bluetooth sul computer.

2.7 Piattaforma ad agenti e Cultura Implicita

2.7.1 JADE

La piattaforma ad agenti da noi utilizzata è stata sviluppata con JADE[1] (Java Agent DEvelopment Framework) [5] versione 3.3, un framework software completamente sviluppato in linguaggio Java. JADE semplifica l'implementazione di sistemi multi-agente attraverso un middle-ware che soddisfa le specifiche FIPA [6] e attraverso un set di tools che supportano il debugging e la fase di sviluppo.

La piattaforma ad agenti può essere distribuita attraverso più macchine (le quali non devono per forza condividere lo stesso sistema operativo) e la configurazione può essere controllata attraverso una GUI remota. La configurazione può essere cambiata a run-time muovendo gli agenti da una macchina ad un'altra, come e quando richiesto. Il solo requisito di sistema necessario è la Java Run Time versione 1.4 o seguente.

L'architettura di comunicazione offre messaggi flessibili ed efficienti; JADE crea e manipola una coda di messaggi di tipo ACL entranti, privati ad ogni agente. Gli agenti possono accedere alla loro coda attraverso una combinazione di vari modi basati su blocking, polling, timeout e pattern matching. È stato implementato un modello di comunicazione completamente FIPA e le sue componenti sono state chiaramente distinte e completamente integrate: protocolli di interazione, sviluppo, ACL, schemi di encoding, ontologie e protocolli di trasporto. Il meccanismo di trasporto, in particolare, è come un camaleonte poiché si adatta ad ogni situazione, scegliendo trasparentemente il miglior protocollo disponibile.

[1] JADE viene utilizzata da un certo numero di società gruppi tecnici, sia membri che non membri di FIPA, come BT, CNET, HHK, Imperial College, IRST, KPN, University of Helsinky, INRIA, ATOS e molti altri. JADE è stata recentemente resa disponibile sotto licenza Open Source.

Vengono attualmente usati la Java RMI, la notifica degli eventi, e IIOP ma possono essere facilmente aggiunti numerosi altri protocolli. La maggior parte dei protocolli di interazione definiti da FIPA sono già disponibili e possono essere istanziati dopo la definizione del comportamento - dipendente dall'applicazione - di ogni stato del protocollo.

Il sistema sviluppato è tuttavia indipendente dalla piattaforma ad agenti utilizzata. Potremmo quindi pensare di installare piattaforme differenti su differenti server, senza per questo compromettere il corretto funzionamento dell'applicazione. Per quanto detto, potremmo quindi avere una situazione simile a quella riportata in Figura 2.8, nella quale sono presenti più server, che supportano differenti piattaforme sviluppate per esempio con JADE, JACK [7], MadKit [8], LEAP [9], ecc.

2.7.2 Cultura Implicita

La Cultura Implicita è la relazione fra un insieme ed un gruppo di agenti che fa in modo che gli elementi dell'insieme si comportino in accordo con la cultura del gruppo. Il supporto della Cultura Implicita diviene utile, se non indispensabile, sia per gli agenti artificiali che per quelli umani. Moltissime volte infatti, per potersi inserire ed amalgamare con un gruppo di persone si è tenuti a comportarsi in accordo con le regole e le usanze di questo gruppo. Un discorso analogo vale certamente anche per gli agenti legati ad utenti mobili: un personal agent deve infatti conoscere il nuovo ambiente in cui opera per poter lavorare al meglio ed ottenere i migliori risultati possibili per il proprio utente.

Per questo motivo si è ritenuto di inserire un modulo di Cultura Implicita anche in *ToothAgent* con il moto "conoscere per apprendere, apprendere per

vincere". Il senso che la frase vuole esprimere è quello che se un agente non conosce l'ambiente che lo circonda, gli agenti che operano attorno a lui, le loro usanze, le loro mosse, ecc. non potrà mai poter competere con loro e di conseguenza non potrà mai ottenere dei risultati soddisfacenti, con ovvio insoddisfaciemeto dell'utente.

Grazie all'utilizzo della Cultura Implicita, ogni agente diviene simile ad un altro, appiattendo così la disparità dovuta alla non conoscenza. Ecco quindi che, per esempio, un agente con il compito di vendere un libro sa come e quanto è stato venduto quel libro in precedenza: può quindi adattarsi (se lo ritiene opportuno) alla media dei prezzi di scambio e porsi ad un livello più concorrenziale. Ed ecco ancora che una persona che è alla ricerca di un compagno con cui condividere un appartamento sa quali sono gli altri utenti che compongono la comunità o che l'hanno composta in precedenza e non perde tempo inutile nell'individuare "chi non esiste". O ancora, un utente interessato a delle particolari informazioni potrebbe farle filtrare dal suo personal agent, in accordo con le scelte fatte in precedenza dagli altri utenti.

Gli utilizzi della Cultura Implicita, così come i servizi possibili offerti da *ToothAgent*, sono moltissimi, a seconda del servizio stesso e di come e cosa chi commissiona l'implementazione di un servizio voglia offrire ai propri clienti. L'idea di base è quella per cui ogni agente interessato all'aiuto da parte del sistema si rivolga al Directory Facilitator (DF) della piattaforma ad agenti e da questo venga indirizzato verso le corrette risorse. Grazie al DF potrà quindi venire a conoscenza degli agenti che lo circondano e di come essi si vogliano comportare nella trattazione/collaborazione. Potrà quindi non solo stimare, accedendo ad un database, i comportamenti passati della comunità ma anche quelli degli attuali membri (Figura 2.9).

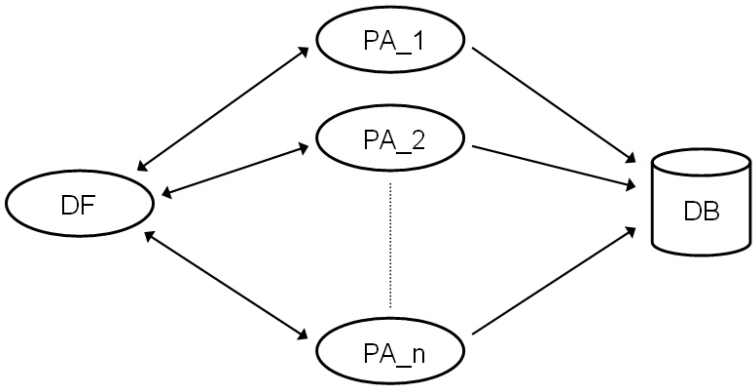


Figura 2.9: Le interazioni fra Persoanl Agent (PA), Directory Facilitator (DF) e DataBase (DB)

Un utente interessato alla vendita del libro "Java Beans" potrebbe così subito osservare, chiedendo al DF, se il libro è effettivamente uno di quelli solitamente scambiati dalla comunità e se in quel momento esistono altri agenti interessati all'articolo. In caso negativo, perchè ostinarsi a ricercare utenti interessati esclusivamente a quello, perdendo del tempo? L'agente sarebbe informato riguardo gli altri libri già scambiati in passato e quelli invece che sono offerti/ricercati al momento. Ecco quindi che il personal agent potrebbe contattare tutti gli agenti interessati all'acquisto di generici libri per Java oppure a libri riguardanti i Beans.

L'esempio è chiaramente espandibile a qualsiasi altro caso di contrattazione o collaborazione e in generale a tutti i processi in cui avvengono delle interazioni fra agenti, con proposte, controproposte, accettazioni e rifiuti.

2.8 Esempi applicativi

Benché, come detto, i servizi possano essere pressoché infiniti, a seconda del luogo in cui risiede il server e dalla tipologia di persone che frequentano quel

luogo, si è deciso di focalizzare l'attenzione e di implementarne quattro a titolo d'esempio.

1. **Compravendita di testi usati**: supponiamo di voler mettere a disposizione degli studenti della facoltà di scienze dell'università di Trento un servizio per la compravendita di libri usati. Un utente interessato a questo tipo di servizio (diciamo alla vendita del libro "Thinking in Java 3rd Edition" di Bruce Eckel), dovrebbe impostare i parametri richiesti fornendo il titolo del libro che desidera vendere, il prezzo desiderato per la vendita e un prezzo minimo fino al quale sarebbe disposto a contrattare. Per contro, l'acquirente dovrebbe invece indicare, oltre al titolo del libro ed il prezzo desiderato, anche un prezzo massimo, oltre il quale non sarebbe più interessato a concludere l'affare. Non conoscendo il titolo del libro che intende acquistare potrebbe per esempio definire alcune parole chiave per la propria ricerca.

2. **Condivisione di automobili**: Una persona potrebbe voler condividere la propria auto per dirigersi verso una particolare zona o città e per questo sarebbe interessata alla ricerca di una o più persone (o per esempio un numero minimo di tre per evitare i blocchi al traffico imposti a causa dello smog) con le quali condividere anche le spese. Pubblica così la propria richiesta indicando ora e luogo di partenza, meta ed eventualmente "prezzo" del viaggio; si mette poi in attesa di ricevere risposte. I parametri per il matching delle preferenze sono in questo caso più semplici rispetto al servizio descritto al punto precedente e non si necessita di particolari contrattazioni o accordi. Il servizio, se utilizzato, sarebbe non soltanto utile ma avrebbe un notevole impatto (anche ambientale!).

3. **Trovare un compagno di viaggio**: in una stazione ferroviaria una persona (chiamiamola A) seduta in sala d'aspetto potrebbe voler cercare un'altra persona (B) con la quale fare amicizia e parlare durante il suo viaggio. B deve ovviamente avere alcune caratteristiche desiderate da A quali: essere maschio o femmina, avere un'età compresa tra un minimo e un massimo, essere interessato ad un certo tipo di argomenti di cui discutere, ma soprattutto deve prendere lo stesso treno che prende A. Per contro, affinché gli agenti possano trovare una corrispondenza ed un accordo, anche A deve avere delle caratteristiche desiderate da B. Come nel caso precedente, una volta trovato l'accordo, i due agenti si scambiano le informazioni dei rispettivi utenti e le memorizzano sul database del server posto nella sala d'aspetto della stazione ferroviaria, rendendole disponibili per una successiva consultazione.

4. **Condividere un appartamento universitario** : molti studenti che frequentano un'università provengono spesso da città lontane, se non da Paesi lontani. Questi studenti, essendo nuovi e non avendo molti amici del luogo, potrebbero essere interessati alla ricerca di un appartamento e di uno o più compagni con cui condividerlo. Poiché spesso questa ricerca può risultare difficile, si può pensare di fornire un servizio ad hoc per agevolare i contatti fra gli studenti. Colui che intende usufruire del servizio, deve dichiarare le proprie preferenze indicando per esempio se sta cercando un ragazzo o una ragazza, i limiti d'età e la nazionalità del compagno desiderato, deve indicare che corso universitario sta frequentando e se ha preferenze riguardo al corso universitario frequentato dal suo compagno; inoltre deve specificare la dimensione dell'appartamento che sta cercando, ovvero il numero di

inquilini che esso può ospitare. L'agente una volta avviato, inizierà la ricerca di altri agenti con caratteristiche simili. Ancora una volta, qualora un accordo venisse trovato, gli agenti si scambierebbero alcuni dati degli utenti proprietari per permettere un successivo contatto telefonico o per e-mail.

Questi sono solamente quattro degli innumerevoli servizi che potrebbero essere messi a disposizione degli utenti. I servizi possono infatti essere offerti da chiunque e in qualunque luogo, senza vincoli. *ToothAgent* potrebbe inoltre essere leggermente modificato per renderlo funzionante e funzionale ad una ed una sola situazione. Si pensi per esempio ad una conferenza a cui possono partecipare decine o centinaia di persone. Individuare una o più persone con le quali confrontarsi e discutere privatamente di argomenti specifici può non essere semplice, specie perché i partecipanti quasi sempre non si conoscono e non hanno mai avuto rapporti in precedenza. Com'è possibile venire incontro alle esigenze di queste persone, interessate alla formazione di una comunità dapprima virtuale e poi reale? *ToothAgent* può chiaramente fare al caso nostro: evitando la configurazione del servizio tramite PC, potrebbe essere fornito ai partecipanti un software ad hoc, funzionante solo per l'occasione e nel contesto descritto. I parametri sarebbero configurabili tramite cellulare/PDA e potrebbero perciò essere modificati in ogni occasione e senza la necessità di un PC. Anche il server sarebbe "staccato" dal mondo (ovvero non sarebbe collegato a nessun altro server che supporta servizi per *ToothAgent*) ed eventualmente potrebbe comunicare solo con gli altri server posti nelle altre aule dove si svolge la conferenza, per garantire all'utente la connessione e la possibilità di osservare i risultati ricevuti. Questi risultati potrebbero contenere i nomi, le e-mail o i numeri di telefono degli altri partecipanti interessati allo

stesso argomento di discussione e ricavati dal personal agent attraverso la collaborazione con altri agenti o con il semplice matching delle preferenze degli utenti. Le risposte sarebbero quindi recuperabili solamente dai server presenti nel luogo della conferenza (poiché essi non sono collegati via internet agli altri server esterni) e questo sarebbe esattamente quello che ci si aspetta, visto che i risultati sarebbero utili solo se ricevuti rapidamente e nel luogo della conferenza.

Per quanto riguarda invece il caso generale, la situazione è leggermente diversa: non appena l'utente con il proprio dispositivo mobile entra nel raggio d'azione del server, i parametri di configurazione vengono immediatamente spediti ad esso, attivando così un nuovo agente che inizia la ricerca di altri agenti interessati allo stesso servizio. Quando vengono individuati uno o più agenti, viene dato il via alla contrattazione/collaborazione. Trovato un accordo che possa soddisfare entrambi (sempre se un accordo viene trovato), i due agenti provvedono a memorizzare le informazioni dell'utente col quale hanno contrattato ed eventuali altri dati sul database del server. L'utente nel frattempo potrebbe aver sospeso la ricerca dei server oppure potrebbe essere uscito dal raggio d'azione dell'access pont; in ogni caso l'agente sarebbe comunque sopravvissuto e avrebbe continuato a svolgere il proprio lavoro.

Quando l'utente desidera osservare se sui server che ha visitato sono presenti delle risposte, non deve fare altro che attivare la ricerca delle risposte sul proprio cellulare/palmare. Non appena viene individuato un server facente parte del sistema che supporta un qualsiasi servizio, viene passata ad esso la lista degli IP dei server visitati. Il server (essendo collegato tramite internet) contatterà uno ad uno tutti i server presenti nella lista passatagli

e recupererà i risultati memorizzati sui rispettivi database. Ricevuti i risultati, li invierà utilizzando la connessione Bluetooth al dispositivo mobile. L'utente a quel punto avrà a disposizione alcuni dati (un sottoinsieme di quelli forniti in fase di registrazione) della persona con la quale il suo agente ha contrattato e potrà contattarla per esempio tramite telefono o e-mail.

La ricezione delle risposte non deve per forza di cose avvenire immediatamente e contattando il server sul quale stanno lavorando gli agenti. Potrebbe per esempio avvenire che l'utente che ha richiesto la ricerca di una persona con la quale parlare durante il tragitto in treno riceva le risposte qualche ora dopo, mentre si trova in università o in biblioteca; o perchè no, uno studente potrebbe ricevere conferma della vendita di un proprio libro appena prima di salire sul treno, quando è seduto in sala d'aspetto, oppure mentre sta passeggiando per la città e passa nelle immediate vicinanze del server dell'agenzia viaggi di cui si è parlato nella sezione 2.4.

Un altro metodo per recuperare i risultati memorizzati sui database consiste nell'inviare al proprio PC la lista degli IP visitati. Una volta fatta questa operazione, utilizzando l'interfaccia grafica messa a disposizione dall'applicazione, è possibile accedere, utilizzando la rete Internet, ai database dei server visitati e scaricare così, in modo del tutto automatico, le risposte eventualmente in essi contenute. Tali risposte saranno visibili sulla stessa interfaccia grafica dell'applicazione.

Capitolo 3

Implementazione

In questo capitolo si discuterà delle scelte d'implementazione adottate e verrà descritto passo passo l'utilizzo del sistema. Verranno quindi trattati i temi inerenti la registrazione on-line dell'utente, la selezione, la configurazione e l'accesso ai servizi, l'utilizzo della Cultura Implicita, il recupero delle risposte pendenti, e l'interazione fra gli agenti.

3.1 Lato PC

3.1.1 Registrazione on line

La registrazione al sistema è il primo passo che ogni utente deve compiere per poter usufruire dei servizi. Essa viene eseguita collegandosi alla rete Internet ed utilizzando un browser qualsiasi.

Sul server centrale, connesso alla rete e quindi ad ogni altro server disposto sul territorio, sono state caricate alcune pagine php che, interfacciandosi ad un database MySQL, permettono all'utente la memorizzazione, l'update e il download delle informazioni in esso contenute.

Il funzionamento: l'utente deve compilare la form inserendo, oltre ai

propri dati personali anche i dati relativi al proprio dispositivo mobile, ovvero l'indirizzo Bluetooth e il numero di cellulare. I dati personali, ben visibili in Figura 3.1, potranno essere utilizzati dai vari servizi per comunicare all'utente con cui si è contrattato alcuni estremi quali, per esempio, il nome, il cognome, l'e-mail, l'età, ecc.

L'utente dovrà inoltre fornire anche una password, che gli verrà richiesta ogni volta che utilizzerà il programma installato sul proprio dispositivo mobile. La password infatti, assieme all'indirizzo Bluetooth del dispositivo dell'utente, sarà necessaria per accedere ai vari servizi (ricerca dei server e ricerca delle risposte ottenute) in sicurezza, evitando quindi l'inconveniente che qualcuno possa accedere alle nostre risposte (per es. in caso di furto del dispositivo); il tipo di protezione e lo scopo sono quindi simili a quello dell'inserimento del codice pin ogni qualvolta accendiamo un cellulare.

L'indirizzo Bluetooth diventa quindi una sorta di login che, utilizzato in coppia con la password, permette all'utente anche di modificare i propri dati personali, come per il numero di cellulare o la password stessa e aggiungere un nuovo dispositivo (fornendo chiaramente il nuovo indirizzo Bluetooth).

Tutte le informazioni fornite dagli utenti vengono memorizzate come detto sul database centrale; i server disposti sul territorio che offrono qualche servizio, accedono a questo database per verificare l'autenticità dei dati ricevuti e per identificare l'utente. Gli utenti registrati, ottengono il diritto di scaricare il software per il PC e il software per il cellulare/PDA (che sono due file jar), e il file di configurazione che contiene le informazioni riguardanti tutti i server e tutti i servizi disponibili (Figura 3.2).

Figura 3.1: Il form di registrazione al sistema

3.1.2 Selezione dei servizi

La selezione e l'impostazione dei servizi viene fatta utilizzando l'applicazione per il PC, un programma scritto in Java che utilizza il file di

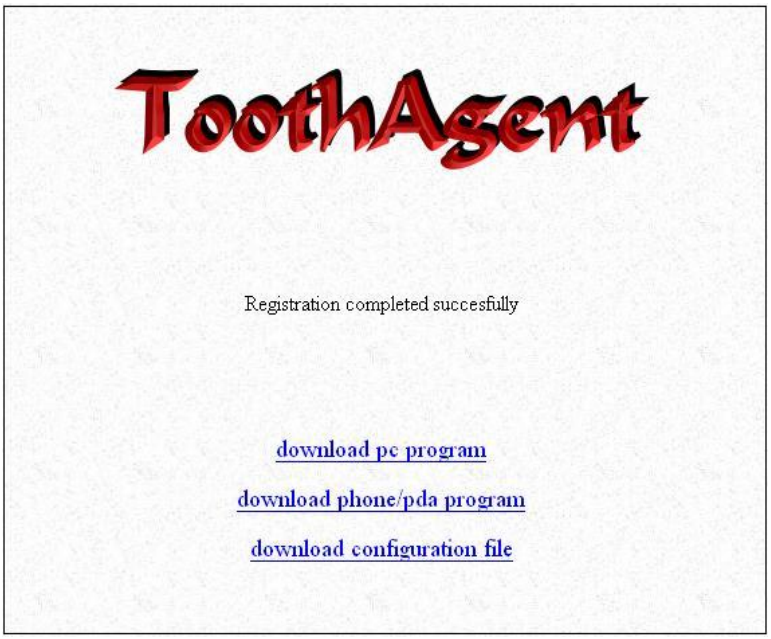


Figura 3.2: La schermata dalla quale è possibile effettuare il download dei programmi e del file di configurazione

configurazione scaricato per la costruzione delle varie videate: dalla lettura e dall'interpretazione di questo file ricava infatti tutti i dati necessari per il settaggio delle varie form da compilare. I dati inseriti e salvati vengono quindi memorizzati in un apposito file e letti nel momento dell'invio al dispositivo mobile.

Il funzionamento: terminata la registrazione e l'installazione del software sul proprio PC, l'utente può iniziare a selezionare i servizi che intende utilizzare. Per fare questo utilizzerà la Java GUI mostrata in Figura 3.3. Utilizzando questa interfaccia, l'utente ha la possibilità di selezionare i server utilizzando alcuni criteri di filtraggio, come per esempio la locazione (si può pensare di avere server collocati in differenti città o in differenti luoghi della stessa città), la tipologia di servizio (si può essere interessati alla ricerca di un libro usato piuttosto che alla ricerca di un compagno con cui condividere un appartamento universitario) o una keyword contenuta nel

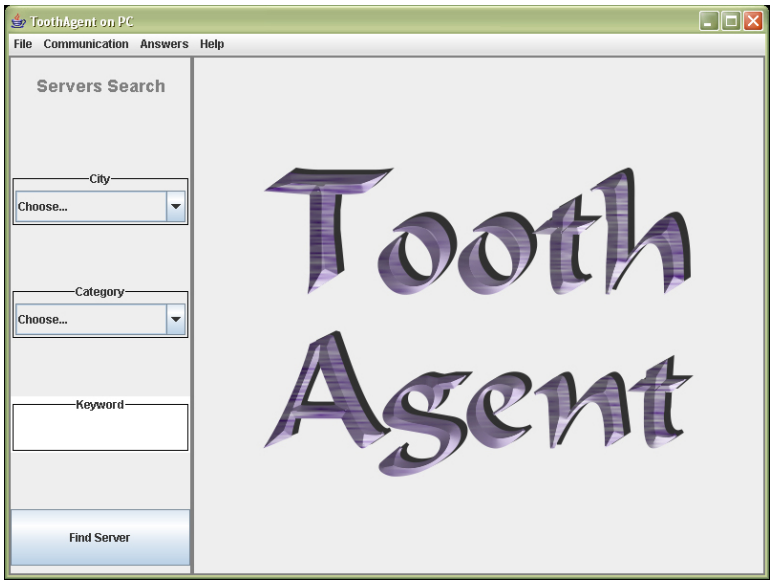


Figura 3.3: La schermata iniziale dell'applicazione per il PC

testo di descrizione del servizio.

Una volta selezionati i parametri della ricerca, la videata mostra tutti i server che possiedono le caratteristiche desiderate dall'utente (in Figura 3.4 per esempio sono mostrati tutti i server presenti nella città di Trento); le informazioni visualizzate sono solo un "riassunto" del servizio; vengono infatti riportati solamente il nome del server, la città e indirizzo della sua collocazione, e le categoria del servizio offerto. L'utente, individuato il servizio di suo interesse, può ora osservare i dettagli cliccando sulla relativa icona; si aprirà così una nuova schermata nella quale, oltre alla descrizione dettagliata del servizio, saranno presenti anche tutti i parametri che dovranno essere settati per la corretta impostazione.

In Figura 3.5 è riportato un esempio di servizio: l'utente in questo caso è interessato alla vendita del libro "Thinking in Java" e richiede per esso un prezzo di 22 euro; l'utente dichiara tuttavia di poter trattare il prezzo e di essere disposto a scendere sino a un limite di 18 euro pur di vendere il libro

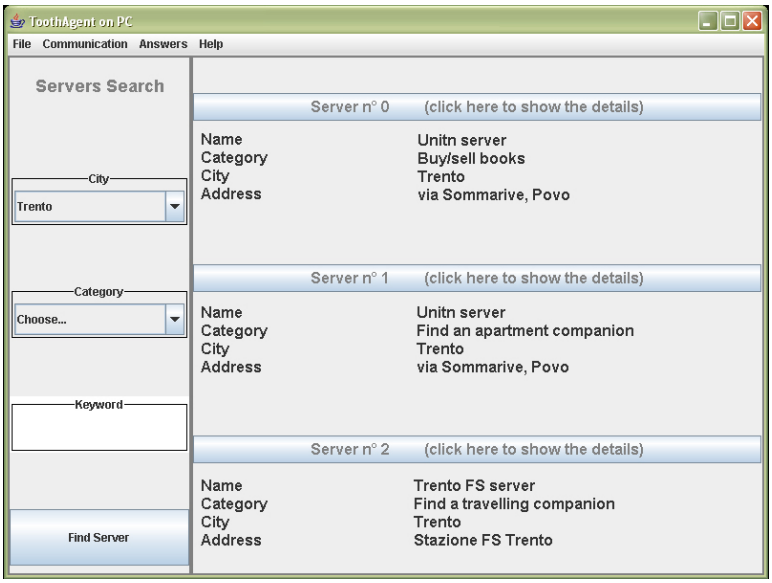


Figura 3.4: La schermata che mostra la lista dei server con le caratteristiche cercate

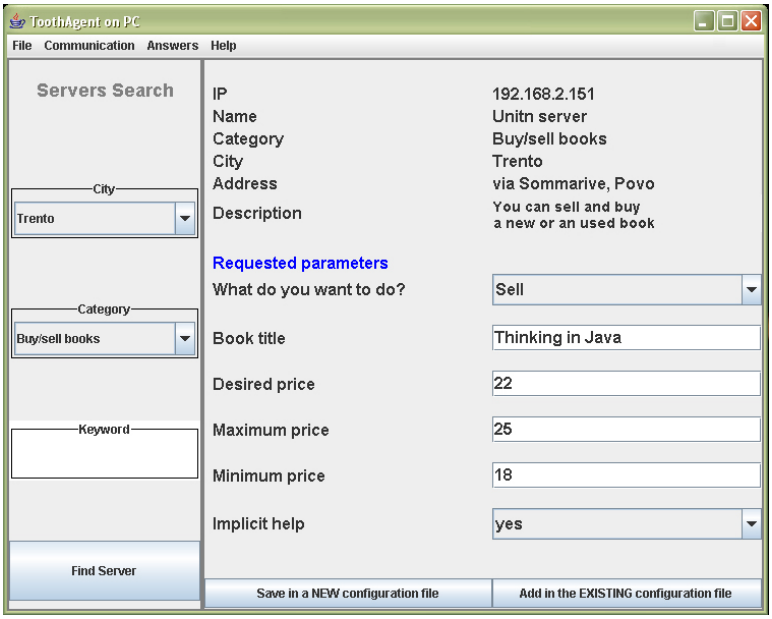


Figura 3.5: La schermata che mostra in dettaglio i parametri richiesti dal server selezionato per l'impostazione del servizio

e di superare l'offerta di qualche altro venditore.

Attivando l'aiuto della Cultura Implicita, egli dichiara inoltre di non

conoscere con esattezza il prezzo di mercato del proprio libro e di volersi per questo far aiutare dal sistema nello stabilire il prezzo migliore per effettuare la vendita, nel caso quello da lui impostato non sia abbastanza concorrenziale.

La preferenza può essere salvata in un nuovo file, cancellando così tutte le precedenti, oppure può essere aggiunta alla lista di quelle già esistenti e precedentemente configuarate. La lista dei servizi viene quindi memorizzata in un file di configurazione XML, il quale verrà inviato tramite connessione Bluetooth al dispositivo mobile. La Figura 3.6 mostra un esempio di file XML generato per il servizio di compravendita di un libro (lo stesso mostrato in Figura 3.5). Da sottolineare come, ancora una volta, il formato non dipenda dal servizio descritto, cosa che implica l'indipendenza dal dominio.

```
<server>
  <ip> 192.168.2.151 </ip>
  <category> Buy/sell books </category>
  <name> Unitn server </name>
  <city> Trento </city>
  <address> via Sommarive, Povo </address>
  <description> You can sell and buy
    a new or an used book
  </description>
  <parameters>
    <param>
      <question> What do you want to do? </question>
      <paramvalue> Sell </paramvalue>
      <inputtype> String </inputtype>
    </param>
    <param>
      <question> Book title </question>
      <paramvalue> Thinking in Java </paramvalue>
      <inputtype> String </inputtype>
    </param>
    <param>
      <question> Desired price </question>
      <paramvalue> 22 </paramvalue>
      <inputtype> double </inputtype>
    </param>
    <param>
      <question> Minimum price </question>
      <paramvalue> 18 </paramvalue>
      <inputtype> double </inputtype>
    </param>
    <param>
      <question> Implicit help </question>
      <paramvalue> yes </paramvalue>
      <inputtype> String </inputtype>
    </param>
  </parameters>
</server>
```

Figura 3.6: Il codice xml creato dal salvataggio dei parametri

3.2 Lato dispositivo mobile

3.2.1 Accesso ai servizi

L'applicazione installata sul dispositivo mobile è programmata in Java e utilizza le JSR-82 che, come riferito nella sezione 1.3 sono le API Bluetooth per Java. Il programma non richiede dispositivi particolarmente sofisticati: basta che questi supportino la tecnologia Java e, ovviamente, la comunicazione Bluetooth.
L'installazione avviene in modo rapido e immediato: dopo aver scaricato il file jar sul proprio PC basta trasferirlo in qualsiasi modo (via Bluetooth, ma anche via IR o USB e/o utilizzando programmi proprietari come Nokia PC suite) sul cellulare/PDA; in pochissimi secondi il dispositivo è pronto e l'utente può iniziare ad utilizzare il programma.

Il funzionamento: per ricercare i server e accedere ai servizi, l'utente non deve fare altro che attivare la comunicazione Bluetooth sul proprio cellulare/PDA e attivare la relativa opzione nel menu del programma. Inizia così una ricerca periodica e continua che prosegue sino al momento in cui l'utente non decide di interromperla.
Il sequence diagram mostrato in Figura 3.7, descrive le interazioni fra il server e il dispositivo mobile dell'utente. Non appena viene individuato un server, il dispositivo mobile stabilisce con esso una connessione e invia il proprio indirizzo Bluetooth (passo 1); il server risponde immediatamente con il proprio indirizzo IP (passo 2); l'applicazione che sta lavorando sul cellulare/PDA verifica se l'indirizzo IP ricevuto fa parte della lista dei server per cui l'utente ha configurato qualche servizio. In caso negativo, il dispositivo mobile interrompe la comunicazione, altrimenti prosegue lo scambio di messaggi inviando, criptata, la propria password (digitata

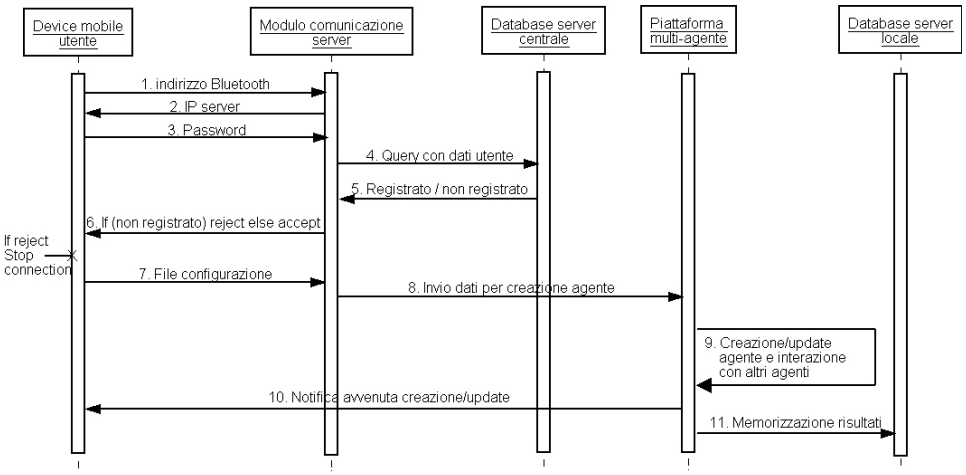


Figura 3.7: Il sequence diagram che descrive il modo in cui avviene l'accesso ai servizi

dall'utente all'inizio della sessione) (passo 3). Il server verifica a questo punto la correttezza della coppia indirizzo Bluetooth - password ricevuta accedendo al database del server centrale (passi 4-5); ancora una volta la comunicazione può interrompersi, in caso di dati errati o mancanti, oppure proseguire. Se il dispositivo mobile riceve l'ok da parte dei server (passo 6), estrapola dal file di configurazione solamente i dati necessari al server in questione e invia questi dati (passo 7). Il server a questo punto può dare vita ad un nuovo agente oppure, nel caso esista già un agente attivo mappatto sullo stesso indirizzo Bluetooth, può aggiornare i servizi che l'agente deve svolgere per conto dell'utente (passi 8 e 9). L'utente riceve quindi notifica sul proprio dispositivo mobile dell'avvenuta creazione o aggiornamento del proprio personal agent (passo 10) e può decidere se interrompere la ricerca di altri server oppure continuarla. In ogni caso l'agente rimane attivo sulla piattaforma e inizia la propria ricerca di risposte utili da fornire all'utente contattando altri agenti con cui contrattare o collaborare. Ottenute le risposte desiderate, l'agente le memorizza sul database del server locale (passo 11) e, a seconda della particolare implementazione del servizio, può

continuare il suo lavoro oppure cessare, deregistrandosi in quest'ultimo caso dalla piattaforma.

3.2.2 Recupero dei risultati

L'utente del sistema, in qualsiasi momento, può attivare sul proprio dispositivo mobile la ricerca dei risultati pendenti. L'applicazione inizierà a questo punto, come avveniva in precedenza per l'individuazione dei servizi disponibili, una periodica e continua ricerca dei server facenti parte del sistema. Non ha importanza che il server individuato sia stato precedentemente visitato dall'utente; in ogni caso, utilizzando la lista degli IP relativa ai server visitati, sarà possibile recuperare le informazioni memorizzate su tali server.
Il sequence diagram riportato in Figura 3.8, mostra lo scambio di messaggi fra il dispositivo mobile dell'utente e il server contattato.

In prima istanza avviene l'invio dell'indirizzo Bluetooth e della password (criptata) dell'utente (passo 1); il server, accedendo al database centrale, controlla la correttezza dei dati ricevuti verificando che l'utente sia registrato al sistema e invia la risposta, affermativa o negativa che sia, al dispositivo (passi 2-4). La comunicazione, che può chiaramente proseguire solo nel caso di accettazione, prevede ora l'invio del file contenente la lista degli IP relativi ai server visitati (passo 5). Il server interrogato dall'utente, provvederà quindi a contattare uno dopo l'altro tutti i server presenti nella lista richiedendo le eventuali risposte presenti nei loro database (passi 6-8). I risultati vengono quindi ritrasmessi al server richiedente (passo 9-10) che provvederà ad inviarli al dispositivo mobile previa verifica della sua presenza (passo 11-13).

Questa tuttavia non è l'unica possibilità: l'utente ha l'alternativa di

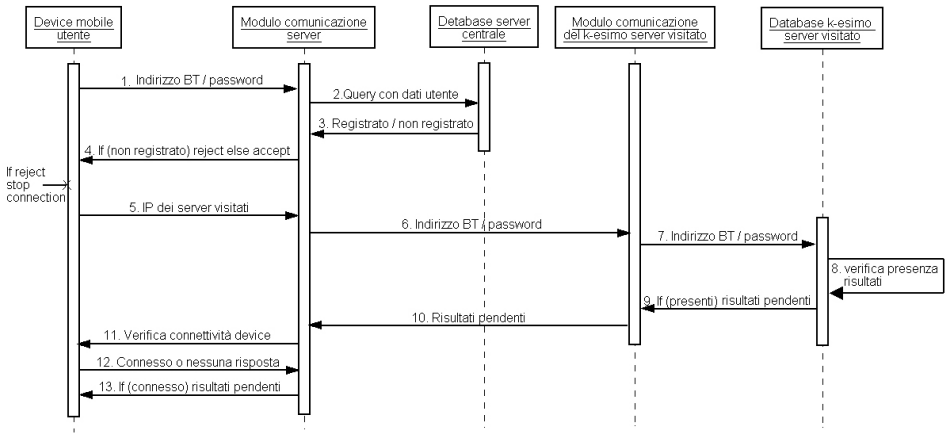


Figura 3.8: Il sequence diagram relativo al recupero dei dati pendenti

recuperare i dati pendenti anche utilizzando il proprio PC. In tale caso è però necessaria una connessione internet. Il protocollo di comunicazione rimane sostanzialmente invariato, tranne per il fatto che l'utente deve trasferire via Bluetooth la lista dei server visitati al computer utilizzando la relativa funzionalità. Prima di iniziare la ricerca vera e propria, l'utente è chiamato ad inserire l'indirizzo Bluetooth e la propria password per i consueti controlli di autenticità. Tale operazione viene fatta utilizzando la schermata riportata in Figura 3.9.

Il sequence diagram che descrive la comunicazione è invece riportato in Figura 3.10. Come detto, l'utente scarica dapprima il file contenente la lista degli IP visitati sul proprio PC utilizzando la relativa funzionalità messa a disposizione dall'applicazione (passo 1) e, dopo aver inserito i propri dati (passo 2), invia la richiesta. Il protocollo è quindi simile a quello precedentemente descritto: dopo l'invio e il controllo dei dati (passo 3), vengono stabilite delle connessioni con tutti i database dei server visitati e attraverso delle opportune query vengono recuperate le risposte pendenti (passi 4-5) che sono successivamente memorizzate nel PC dell'utente e rese

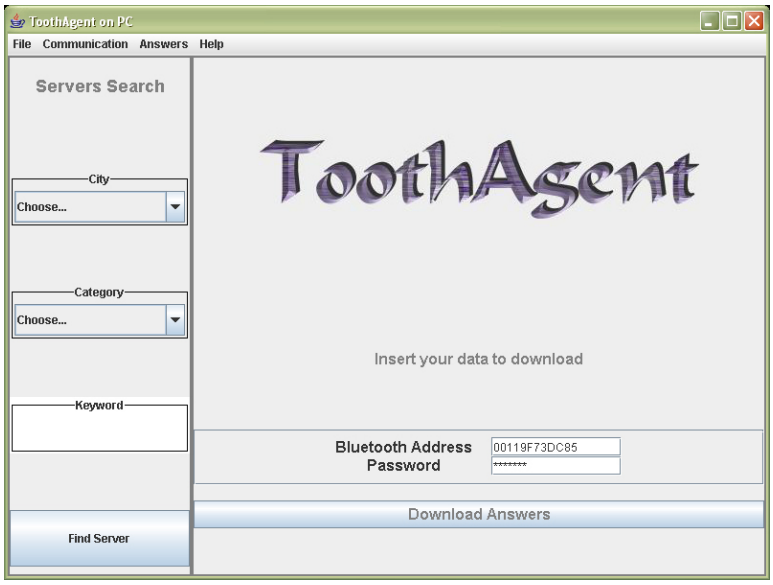


Figura 3.9: La schermata dell'applicazione che permette l'inserimento dei dati dell'utente necessari per il recupero delle risposte dai vari server visitati

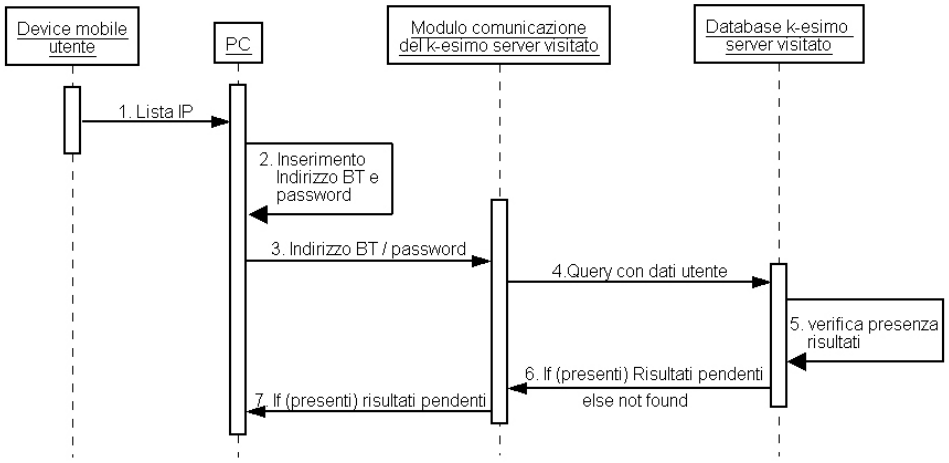


Figura 3.10: Sequence diagram che descrive le interazioni fra i vari componenti del sistema al fine di recuperare i dati pendenti

disponibili per la visualizzazione (passi 6-7).

In Figura 3.11 è possibile osservare le risposte scaricate dalla rete in seguito alla ricerca effettuata su tutti i server precedentemente visitati. Come si può notare, vengono fornite delle informazioni compatte ma essenziali

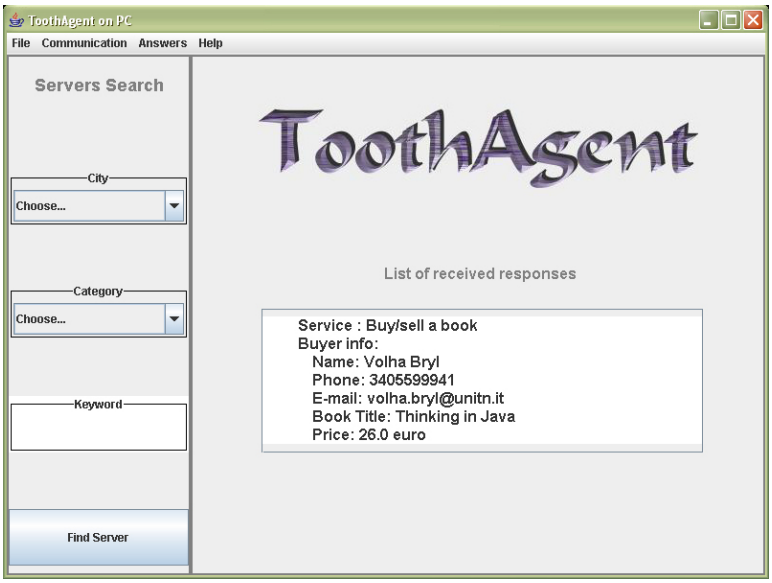


Figura 3.11: La schermata finale della ricerca delle risposte in cui vengono riportati tutti i risultati delle contrattazioni/collaborazioni avvenute sui server visitati

affiché l'utente possa contattare la persona interessata all'acquisto del libro. Oltre al prezzo concordato dagli agenti, sono infatti riportati il nome ed il cognome dell'acquirente, accompagnati dal suo numero di cellulare e dall'e-mail. Chiaramente, le stesse informazioni (ma riguardanti il venditore) verranno ricevute anche dal compratore.

3.2.3 Un esempio completo

In questa sezione mostreremo graficamente il funzionamento di *ToothAgent* ed in particolare l'applicazione per i dispositivi mobili. Le figure seguenti mostrano passo dopo passo una delle simulazioni eseguite, quella per la compravendita di un libro.

Dopo aver installato il programma sul cellulare/PDA (Figura 3.12 a), si entra nella schermata principale del programma dalla quale si può accedere ai vari sotto menu: servizi, risposte, server visitati, impostazioni,

connessione (Figura 3.12 b). Accedendo al menu “servizi”, si presentano due alternative: visualizzare i servizi attivi (già scaricati sul cellulare) oppure scaricare il file di configurazione dal PC (Figura 3.12 c).

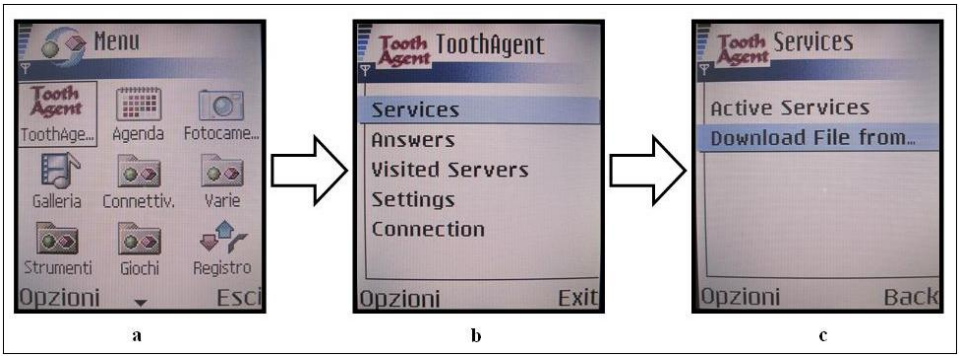


Figura 3.12: Download del file di configurazione

Scegliendo quest’ultima alternativa, e attivando contemporaneamente la connessione Bluetooth sul computer, inizierà il trasferimento e la successiva memorizzazione del file (Figura 3.13 a). A questo punto si può far partire il servizio vero e proprio, ovvero la ricerca dei server per l’attivazione dei propri personal agent (Figura 3.14 a); prima di fare questo però si dovranno impostare le proprie preferenze inserendo la password scelta in fase di registrazione (Figura 3.13 b) e, opzionalmente, il tempo di attesa tra una ricerca e la successiva (il cui scopo sarà illustato nella sezione 3.4) (Figura 3.13 c). La connessione e la conseguente ricerca dei server è quindi ora attiva ma si avrà sempre la possibilità di interromperla utilizzando l’apposito menu nell’area “connessione”.

Nell’attesa di ricevere notifica dell’avvenuta creazione dei propri personal agent su qualche piattaforma (Figura 3.14 c), cosa che avviene ad ogni nuova creazione o aggiornamento, si può tranquillamente utilizzare il cellulare/PDA per altri scopi, con l’accortezza però di mantenere il programma attivo in background. Per esempio, come mostrato in Figura

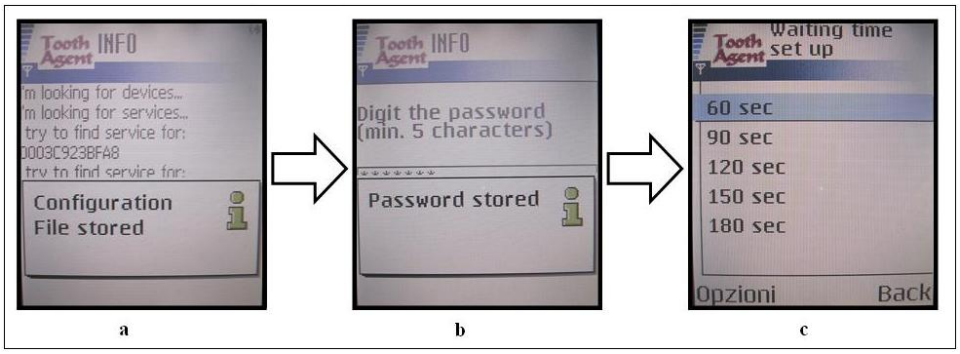


Figura 3.13: Impostazione della password e del tempo di attesa fra una ricerca e la seguente

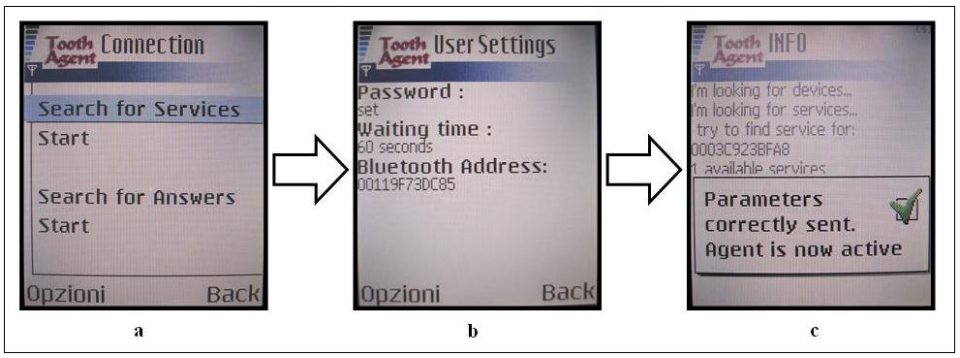


Figura 3.14: Avvio della ricerca dei server per lanciare i propri agenti

3.14 b, l'utente continua il proprio lavoro navigando nel menu del programma e posizionandosi sulla schermata relativa ai parametri impostati .

Quando si desidera ricercare delle risposte pendenti, si entra come in precedenza nel menu "connessione" attivando l'opzione "ricerca risposte" (Figura 3.15 a). Il funzionamento è analogo a quello descritto sopra: si potrà in qualsiasi momento interrompere la ricerca e si potrà utilizzare il dispositivo per altri scopi mantenendo il programma attivo in background. Ogni nuova ricezione di risposte sarà tempestivamente segnalata con un breve messaggio di informazione (Figura 3.15 b); i dati verranno salvati in memoria e saranno visibili nell'area "risposte" (Figura 3.15 c). Da quest'area sarà possibile inviare i dati ricevuti al PC e cancellare le risposte lette.

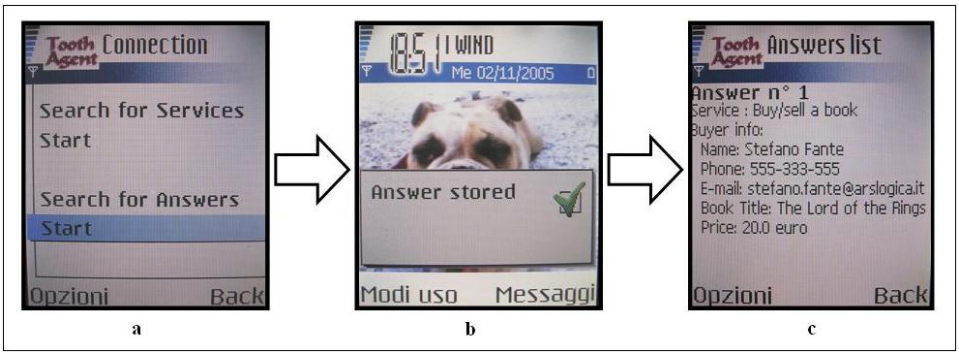


Figura 3.15: Ricezione, memorizzazione e visualizzazione delle risposte

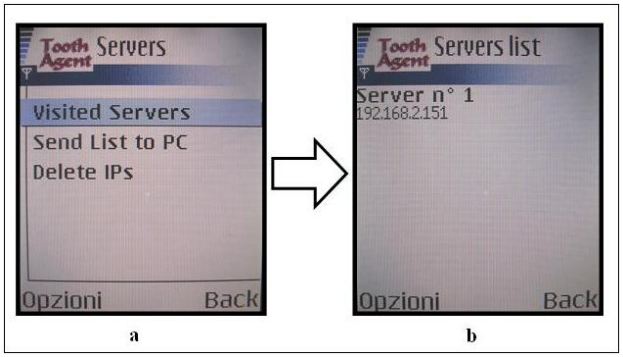


Figura 3.16: Visualizzazione dei server visitati

Da ultimo, nell'area "server visitati" sarà possibile osservare la lista degli IP relativi ai server contattati e inviare tale lista al computer per il recupero delle risposte pendenti in modalità PC (Figure 3.16 a-b).

3.3 Lato server

3.3.1 Cultura Implicita

Rifacendosi al modello presentato in [15] e mostrato in Figura 3.17, si è implementata una versione semplificata del modulo per la gestione della Cultura Implicita. Come si può osservare, la Figura 3.17 mostra il modello SICS (System for Implicit Culture Support) già presentato in [13] . Un SICS

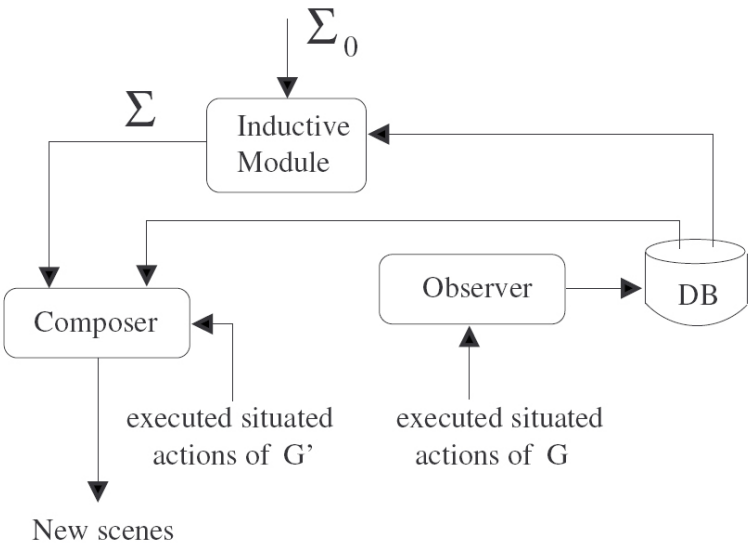


Figura 3.17: Il SICS presentato in [15]

generico consiste di tre componenti: un osservatore (observer), un modulo induttivo (inductive module) e un compositore (composer). L'osservatore memorizza le azioni eseguite da un gruppo di agenti G in modo da renderle disposnibili in seguito per gli altri componenti. Il modulo induttivo utilizza queste azioni per produrre delle teorie culturali Σ per G. Da ultimo, il compositore, usando la teoria Σ e le azioni, manipola l'ambiente creato da un insieme di agenti G' in modo tale che le azioni aspettate siano di fatto azioni culturali rispetto a G. Come risultato, gli agenti di G' eseguono (in media) delle azioni legate alla cultura di G, e così il SICS produce un fenomeno di Cultura Implicita.

In un sistema multi-agente, un SICS può essere sia una capacità generale dell'intero sistema, sia un capacità di un singolo agente. Nel primo caso, il SICS osserva tutti gli agenti che agiscono nel sistema e che manipolano l'ambiente. Nel secondo caso invece, il SICS viene applicato a quello che l'agente è capace di fare, ovvero a quello che è in grado di osservare e di cambiare, cioè la parte di ambiente e gli agenti con cui interagisce. Il SICS,

generale o specifico che sia, influenza l'intero sistema.

In *ToothAgent* è stata utilizzata la seconda opzione nella quale il SICS è una capacità del singolo agente. Per ottenere questo si è inserito il SICS nel Directory Facilitator (DF) della piattaforma, ovvero nell'agente che gioca il ruolo fondamentale nelle interazioni, estendendo le sue funzionalità rispetto a quanto riportato nelle specifiche FIPA. Il DF, secondo le specifiche FIPA, è infatti un agente presente di obbligo nella piattaforma; il suo compito è quello di fornire una sorta di pagine gialle agli altri agenti presenti. Ogni agente che desidera far conoscere i propri servizi agli altri è tenuto a registrarsi presso il DF ed a fornire una descrizione dei servizi. Un agente così, richiedendo al DF un determinato servizio, verrà subito indirizzato verso chi lo offre.
Grazie al SICS, il DF può produrre una relazione di Cultura Implicita fra l'agente e gli agenti che hanno richiesto in precedenza le informazioni e fornire un'informazione filtrata che può influenzare la preferenza dell'agente. La presenza di SICS diminuirà quindi la quantità di richieste perse, nulle o mal formate, migliorando così le prestazioni del sistema, le interazioni e il grado di soddisfacimento dell'utente.

Accadrà quindi che un agente creato per la vendita di un libro, non si limiti a ricercare dei compratori solamente in base ai parametri inseriti dal proprio utente, ma chieda informazioni al sistema sul prezzo medio a cui solitamente quel libro viene venduto; oppure interroghi il sistema per conoscere tutti gli agenti interessati a libri differenti ma con soggetto attinente al suo, per poter eventualmente proporre loro l'acquisto.
In questo caso, il sequence diagram dell'interazione fra agente e sistema è leggermente diverso da quello illustrato in Figura 3.7. Dopo l'invio dei dati per la creazione dell'agente avvengono infatti due operazioni svolte

rispettivamente dall'osservatore e dal compositore: il primo momorizza i dati ricevuti sul database, mentre il secondo, utilizzando questi dati, propone all'agente i nuovi parametri per la configurazione del servizio calcolati sulla base delle esperienze passate e sulla base della situazione attuale della piattaforma (passi 9-10 in Figura 3.18).

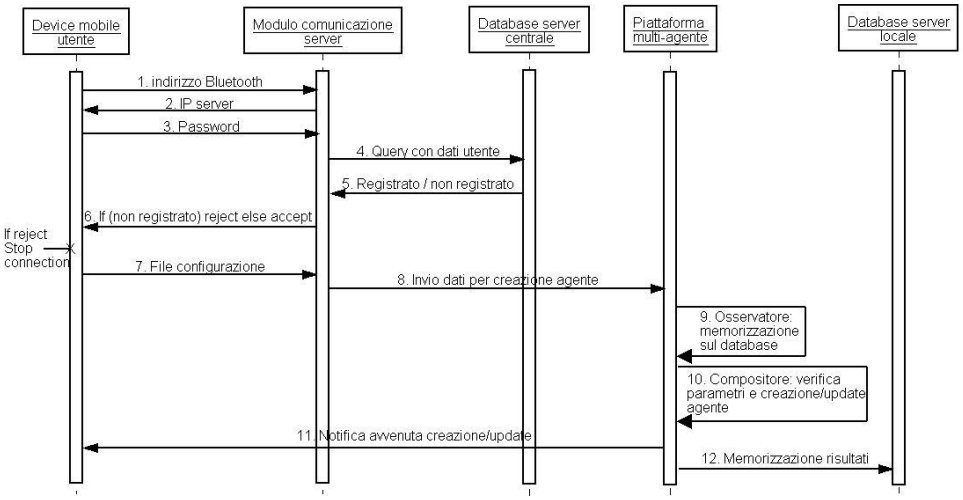


Figura 3.18: Il sequence diagram che descrive il modo in cui avviene l'accesso ai servizi

3.3.2 Interazione degli agenti

L'interazione degli agenti include due fasi: l'elaborazione delle richieste dell'utente e la negoziazione e/o la collaborazione con gli altri agenti.
Ogni servizio, per come è stato pensato il sistema, implementa la propria e personale interazione, indipendentemente dalla piattaforma su cui viene attivato. Potrebbe per esempio accadere che un agente, trovata la prima risposta utile alla richiesta dell'utente, cessi il proprio servizio e, nel caso questo fosse stato il solo, cesserebbe la sua esistenza, deregistrandosi dalla piattaforma. In altri casi invece l'agente potrebbe continuare a vivere e lavorare per un periodo di tempo prestabilito (1 minuto come 30 giorni),

concluso il quale terminerebbe anche il suo "lavoro".

Per quanto riguarda ad esempio il servizio per la compravendita di libri usati, è stato posto un limite di vita dell'agente di 10 giorni, scaduti i quali cesserà la ricerca di altri utenti che vogliono acquistare o vendere il libro cui l'utente è interessato. Il protocollo di contrattazione, anch'esso libero e gestibile da chi realizza il servizio, è stato suddiviso in più fasi; un agente interessato per esempio all'acquisto del libro, diciamo "Thinking in Java" con un prezzo compreso tra 15 e 25 euro, ricerca altri utenti interessati alla vendita dello stesso libro. Trovati uno o più venditori, l'agente richiede il prezzo "di base d'asta" che essi desiderano per quel libro; le offerte ricevute vengono valutate dall'agente, il quale, sulla base delle richieste dell'utente, chiede la possibilità di uno sconto. Gli agenti venditori, qualora lo ritengano opportuno, propongono un nuovo prezzo ribassato oppure confermano il prezzo precedente: questo avviene fino a quando i due agenti non arrivano ad un accordo. L'agente compratore valuta quindi la migliore proposta ricevuta e conclude la trattazione scambiando i propri dati con il miglior venditore.

Altri servizi, quali possono essere ad esempio la ricerca di un compagno con cui condividere un appartamento o la ricerca di un compagno di viaggio, possono non richiedere alcuna contrattazione, ma solamente una verifica delle caratteristiche e degli interessi dell'utente. In questi casi, la collaborazione si può dire conclusa quando si incontrano due o più agenti con caratteristiche simili, compatibili con i desideri degli utenti.

Altri servizi ancora, potrebbero non richiedere affatto l'intervento diretto degli agenti, ma solo una loro semplice attesa dei risultati che verranno forniti da terze fonti; queste potrebbero essere per esempio informazioni riguardanti l'andamento del prezzo di un'azione, il ritardo o la soppressione

di un treno, una proposta commerciale, una promozione speciale per qualche articolo in vendita in qualche particolare negozio, ecc. In tutti i casi e per ogni servizio, l'architettura studiata ed implementata si rivela tuttavia efficiente e garantisce i risultati attesi, poiché gli agenti sono in grado di comportarsi in vari modi, a seconda di come la situazione lo richiede.
Chi implementa i servizi può infatti configurare la competitività così come la collaborazione degli agenti in vario modo. Ogni agente si distinguerà quindi da un altro solo in base ai parametri forniti dall'utente del sistema. Sarà lui in definitiva, attraverso l'inserimento dei dati richiesti, a determinarne la "natura": è pensabile per esempio un servizio in cui venga richiesto all'utente di indicare il grado di collaborazione che dovrà avere il suo agente; questo parametro influenzerà il comportamento dell'agente e determinerà lo scambio finale di informazioni con gli altri agenti.

3.4 Testing sui servizi implementati

ToothAgent è stato testato su due tipi differenti di device mobile, facenti comunque parte della categoria "telefono cellulare". I dispositivi utilizzati sono stati il Nokia 6260 e il Nokia 6630: su entrambi l'applicativo ha dimostrato di funzionare correttamente. La parte PC e server è stata testata utilizzando il dongle Bluetooth Tecom BT3030. In ambienti aperti il raggio di ricezione si aggirava attorno ai 12-15 metri (limite massimo dovuto ai cellulari, non al dongle che è invece di categoria 1 ed ha un raggio d'azione di 100 metri), mentre in ambienti chiusi, con pareti ed ostacoli, il raggio non superava gli 8-10 metri.

Alcuni telefonini, benché si affermi che la tecnologia Bluetooth necessiti di pochissima energia, potrebbero mostrare (come in effetti fanno) un consumo eccessivo della batteria quando il dispositivo Bluetooth è

attivato. Alcune case produttrici, per evitare questo inconveniente, lo fanno disattivare automaticamente se inutilizzato, dopo qualche minuto. I cellulari della Nokia da noi testati non presentano questo inconveniente, ma si è pensato tuttavia di permettere al cliente finale di customizzare il sistema come meglio ritiene opportuno ed utile: vediamo come. Nel sistema implementato, la ricerca di nuovi dispositivi e di nuovi server su cui lanciare i propri agenti o da cui scaricare le proprie risposte pendenti, viene effettuata di default ogni 60 secondi; per ridurre lo spreco energetico si è pensato di introdurre un parametro, a discrezione dell'utente, che prolunghi il tempo di inattività tra una ricerca ed un'altra: l'utente ha così la possibilità di dilatare il tempo di attesa fino ad un massimo di 180 secondi. Questo fatto, positivo dal punto di vista del risparmio energetico (e i test lo hanno dimostrato), potrebbe però apportare degli inconvenienti: aumenta infatti la possibilità di entrare ed uscire dal campo di ricezione di un server senza tuttavia mettersi in contatto con esso e senza quindi attivare i propri personal agent e/o scaricare le proprie risposte pendenti. Questa scelta, come detto, viene lasciata all'utente, che configurerà il dispositivo anche in base alle proprie abitudini o ai propri spostamenti: un utente che passa gran parte del suo tempo nella biblioteca della facoltà per esempio, potrebbe non essere interessato alla ricerca continua di nuovi server, poiché sa che ne esiste solo uno nelle vicinanze e da quello si limita a recuperare le proprie risposte pendenti. Se l'utente vuole essere tempestivamente informato di nuove risposte ricevute, potrebbe quindi dilatare il tempo di attesa fra una ricerca e la successiva ai 180 secondi, senza che questo pregiudichi la ricezione.

Il sistema è stato testato in ambiente universitario ed ha prodotto dei

risultati soddisfacenti. Non è stato tuttavia possibile, a causa del ridotto numero di dispositivi mobili in nostro possesso, un testing più completo e di più ampio raggio, che includesse un gran numero di comunicazioni wireless. Nonostante il ridotto numero di cellulari e PDA, abbiamo comunque simulato varie situazioni, due delle quali sono visibili in Figura 3.19:

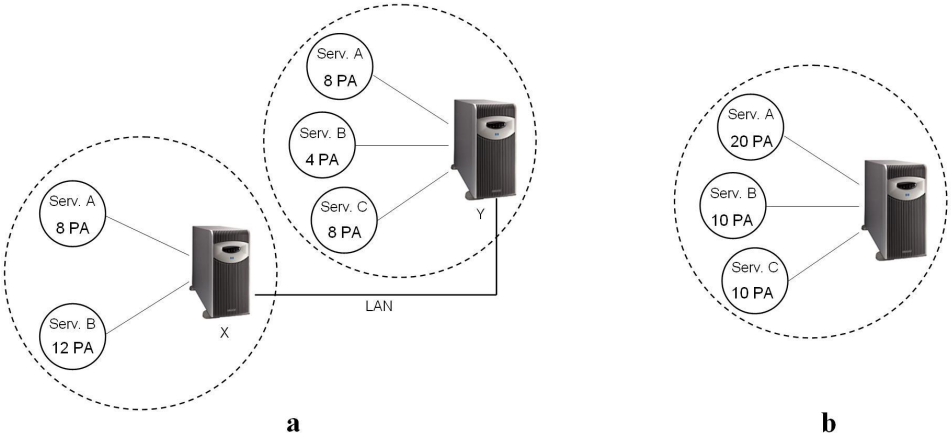


Figura 3.19: Due dei test effettuati sul sistema

- nella prima (Figura 3.19 a) abbiamo creato 40 agenti e li abbiamo lanciati contemporaneamente su due piattaforme (che chiameremo X ed Y) che supportavano tre servizi (i primi tre descritti nella sezione 2.8, che indicheremo nell'ordine con A, B e C). La piattaforma X offriva due servizi e gestiva 20 agenti: di questi, 8 erano interessati al servizio A e 12 erano invece interessati al servizio B. La piattaforma Y supportava invece tutti e tre i servizi e gestiva i rimanenti 20 agenti: di questi, 8 erano interessati al servizio A, 4 erano interessati al servizio B e i restanti 8 erano interessati al servizio C;

- la seconda situazione (Figura 3.19 b) ha visto invece l'interazione di tutti i 40 agenti sulla stessa piattaforma: 20 agenti interessati al servizio A, 10 al B e gli ultimi 10 al C.

Ovviamente questi agenti non sono stati creati attraverso lo scambio di informazioni con i dispositivi mobili (in ogni simulazione solo 2 sono stati creati in questo modo): lo scambio di dati è stato solo simulato, in modo da creare l'agente vero e proprio. I risultati sono stati però quelli attesi dimostrando come, a meno di problemi non constatati di comunicazione wireless, il sistema funzioni correttamente e sia in grado di supportare un gran numero di utenti.

Le prestazioni del sistema, sono state ovviamente anche testate con carichi minori. Il tempo con il quale gli agenti arrivavano ad un accordo era tuttavia pressoché identico in tutti i casi, lasciando supporre che, se qualche problema dovesse insorgere, non dipenderebbe certo dal numero di agenti presenti sulla piattaforma. Riteniamo infatti che 40 agenti su di una sola piattaforma sia un numero abbastanza elevato, considerando il carattere localizzato dell'applicazione e il tempo di vita (limitato) degli agenti.

3.5 Strumenti e tools

- Il sistema è stato sviluppato sul sistema operativo Windows XP SP2, poiché, come vedremo in seguito, si necessita dello stack Bluetooth Microsoft su cui è sviluppato BlueCove.

- Come dongle USB Bluetooth si è utilizzato un Tecom BT3030 (Figura 3.20). In questo modo abbiamo abilitato il PC a comunicare, attraverso un'interfaccia USB e con una connessione wireless, con altri dispositivi Bluetooth. BT3030 supporta la trasmissione di dati ed è compatibile con le specifiche standard Bluetooth 1.1; è un dispositivo di classe 1 e permette quindi una potenza fino

a 20 dBm che supporta operazioni in un range di massimo 100 metri. Maggiori info possono essere trovate sul web al sito *http://www.tecomproduct.com/bt3030.htm*

Figura 3.20: Il dongle Tecom BT3030 utilizzato per i test

- Per lo sviluppo delle MIDlet è stato utilizzato il tool della SUN *WTK 2.2* (Wireless ToolKit) che a sua volta contiene un set di tools per la creazione di applicazioni Java che possano essere installate su un device compatibile con le specifiche della tecnologia Java per il wireless. *WTK 2.2* contiene, fra l'altro, numerose utilità e un emulatore di dispositivi mobili. Per maggiori informazioni consultare il sito: *http://java.sun.com/products/sjwtoolkit/download-2_2.html*.

- Per l'utilizzo del Bluetooth attraverso Java è stato utilizzato *BlueCove* [3], un'applicazione nata da un progetto open source sviluppato da Intel. *BlueCove* non implementa le intere API Java JSR-82, ma solo alcune delle funzioni. Per la traduzione dei comandi fra Java e C++ vengono usate la API Windows Socket e JNI (Java Native Interface). Le buone notizie derivano dal fatto che entrambe le parti Java e C++ sono open source e il codice può essere scaricato e osservato dall'utente. Il progetto *BlueCove* non è ancora stato ultimato e l'attività totale, al momento della scrittura, è stimata essere arrivata all'87.42%. Altre

funzionalità dovranno quindi ancora essere introdotte. Maggiori dettagli su *http://sourceforge.net/projects/bluecove/*.

- L'SDK (Software Development Kit) per lo sviluppo dell'applicazione J2SE è stato *Eclipse 3.1*. *Eclipse* è una comunità open source i cui progetti sono focalizzati a piattaforme estensibili per lo sviluppo e a frameworks per la costruzione di software. *Eclipse* fornisce dei tools estensibili e frameworks per fissare il ciclo di vita di sviluppo del software, includendo il supporto per la modellazione, un ambiente di sviluppo per i linguaggi Java, C/C++ e altri, per il testing e le performances, ecc. Maggiori informazioni e dettagli possono essere trovati su *http://www.eclipse.org/*.

- I dispositivi mobili con cui abbiamo testato il sistema sono stati il *Nokia 6260* (Figura 3.21) e il *Nokia 6630* (Figura 3.22). Per dettagli e per la caratteristiche dei due cellulari è possibile consultare i sito della Nokia agli indirizzi *http://www.nokia.it* oppure *http://www.nokia.com*.

Figura 3.21: Nokia 6260

- La piattaforma multi-agente utilizzata è stata *JADE* (Java Agent

Figura 3.22: Nokia 6630

DEvelopment framework) versione 3.3. *JADE* è una piattaforma open source per applicazioni peer-to-peer basate su agenti totalmente sviluppata in Java. Grazie ad un middle-ware compatibile con le specifiche FIPA e grazie a un set di tool grafici che supportano il debugging e le fasi di sviluppo, *JADE* semplifica l'implementazione di sistemi multi-agente. *JADE* è un software gratuito distribuito da TILAB, il proprietario del copyright; come detto è open source sotto i termini della LGPL (Lesser General Public Licence). Attualmente i membri del progetto sono 5: oltre a TILAB, collaborano infatti anche Motorola, Whitestein Technologies AG., Profactor GmbH, e France Telecom R&D. L'ultima versione di *JADE* è quella da noi adottata, ed è stata rilasciata a marzo 2005. Maggiori dettagli sono disponibili consultando il sito *http://jade.tilab.com*.

Conclusioni e Sviluppi futuri

L'obiettivo di questo lavoro è stato quello di studiare, implementare e verificare un sistema multi-agente in grado di supportare delle comunità virtuali mobili e agevolarne l'interazione tra i membri. In particolare ci si è prefissati di costruire un sistema dinamico e scalabile, capace di evolvere e di ampliarsi nel tempo, senza necessità di ulteriori interventi o modifiche.

Oltre al carattere di assoluta indipendenza dalla piattaforma e dai servizi offerti, prerogativa del sistema è quella di essere facilmente accessibile e con tecnologie ampiamente diffuse. Si è deciso quindi di adottare dispostivi mobili dotati di tecnologia Bluetooth e programmabili in Java.

Il sistema è stato pensato per supportare il concetto di località, ovvero consentire l'accesso solo a quegli utenti che frequentando un deteminato ambiente sono interessati ad usufruire di specifici servizi.

Benché in letteratura sia possibile trovare altri sistemi simili a *Toothagent*, tuttavia la maggior parte di essi non supportano l'interazione fra gli utenti e si limitano alla ricerca statica di informazioni, attingendo da svariate basi di dati o da internet, spesso aiutando l'utente solo con semplici indicazioni turistiche o affini. Tali sistemi generalmente richiedono l'utilizzo di tecnologie di comunicazione non ancora ampiamente diffuse come per esempio il GPS, installato in molte automobili ma non ancora sui cellulari o sui PDA.

Al di là della creazione di nuove comunità virtuali e del supporto ai suoi partecipanti, obiettivo di questo lavoro è stato quello di colmare la mancanza di applicazioni dove agenti software personali interagiscono all'interno di comunità virtuali per soddisfare i bisogni dei loro utenti; tutto questo inserito in un sistema dinamico ed eterogeneo, capace di supportare potenzialmente un'infinita varietà di servizi.

I test effettuati (benché preliminari a causa del limitato numero di dispositivi mobili in nostro possesso) hanno fornito risultati soddisfacenti e in linea con le aspettative iniziali, sottolineando come le scelte della piattaforma ad agenti e della comunicazione wireless siano state corrette ed appropriate.

Come futuro sviluppo si è pensato di rendere i personal agent mobili, ovvero in grado di migrare da un server all'altro e in grado di seguire quindi l'utente nei suoi spostamenti: si avrebbe così una corrispondenza uno-a-uno fra utenti ed agenti riducendo in maniera consistente il numero di agenti presenti sulle piattaforme e di conseguenza il carico di lavoro.
I test potrebbero essere effettuati nelle vie di Trento centro coperte dalla rete wireless grazie al progetto WILMA: alcuni server connessi tra loro senza l'ausilio di fili potrebbero quindi ospitare dei servizi che verrebbero utilizzati dagli utenti che frequentano i bar o i locali della zona o che semplicemente passeggiano per le strade o fanno shopping.

In ogni caso i test non si concluderanno con la stesura di questo lavoro; la stima delle prestazioni del sistema sarà meglio analizzata nei prossimi mesi, quando si inizierà presso ArsLogica il processo di re-ingegnerizzazione per trasformare questo prototipo in un vero e proprio software distribuibile.

Appendice A

Dispositivi mobili

Come detto nella capitolo 2 abbiamo voluto sviluppare un sistema che fosse usabile da una grande quantità di utenti e non limitare quindi l'utilizzo ad una ristretta cerchia di utenti che possiedono dei dispositivi mobili particolari o troppo sofisticati.

Telefoni cellulari

Figura A.1: 4 moderni cellulari

Il numero di cellulari venduti è in continuo aumento: si stima per esempio che in Italia vi siano più di due cellulari per ogni persona. Il telefono cellulare permette agli utenti di ricevere chiamate ed essere raggiungibili in ogni punto del mondo. Oltre a questa caratteristica, che inizialmente era l'obiettivo più

alto, ai telefoni cellulari sono state aggiunti molti nuovi servizi quali per esempio i messaggi di testo, la gestione della rubrica, il calendario, i giochi, le fotografie, la riproduzione del file musicali, ecc. Fino a qualche anno fa, era impensabile poter installare una propria applicazione sul proprio telefonino, ma oggi grazie alle nuove tecnologie (vedi Java) è possibile installare software di terze parti, magari scaricandolo direttamente dalla rete Internet. Nel 2002 solamente l'11% dei telefonini sopportavano Java, mentre si stima che nel 2007 la percentuale salirà fino al 74%. Ad oggi 580 milioni di telefonini e palmari sono abilitati Java (fonte *http://www.it.sun.com*)

PDA

Figura A.2: 2 moderni palmari

Un PDA (Personal Digital Assistance) è un piccolo computer tascabile che permette all'utente di gestire i contatti, l'agenda, memorizzare alcune note personali, ecc., ma può anche includere moltissime altre applicazioni per esempio un web browser o un media player. Di solito posseggono una piccola tastiera o un sistema di input basato su touchscreen. Negli ultimissimi anni, la differenza fra PDA e telefono cellulare si è andata via via assottigliando, tant'è che oggi esistono molti modelli che inglobano l'una e l'altra cosa. Solitamente tuttavia i PDA posseggono delle caratteristiche superiori ai

telefoni cellulari, come per esempio uno schermo più grande, una memoria espandibile e un processore più potente.

Appendice B

J2ME and Bluetooth

Per permettere l'implementazione del Bluetooth sul maggior numero di device possibili, agevolandone la diffusione, è stata progettata e rilasciata una specifica aperta e senza royalty dal Bluetooth SIG che ne definisce sia i livelli hardware, sia quelli software.

Un comportamento ben diverso è invece stato adottato dai produttori dei moduli Bluetooth che forniscono delle apposite SDK per interfacciarsi con i propri moduli, vincolando in questo modo gli sviluppatori all'uso di API proprietarie per i loro rispettivi chip-sets. Una diretta conseguenza è che le applicazioni Bluetooth costruite utilizzando queste API non sono portabili su device e piattaforme diverse, nonostante la tecnologia sia basata su delle specifiche standardizzate. In realtà però queste specifiche sono di tipo funzionale nel senso che descrivono come i dispositivi Bluetooth si comportano e come interagiscono tra di loro attraverso i profiles, tralasciando il modo con cui questi devono essere programmati o come un'applicazione deve utilizzare le funzionalità di questi dispositivi ed i canali di comunicazione.

Per risolvere questi problemi di compatibilità, che ostacolavano la

creazione di applicazioni, si è resa necessaria la creazione di un set di API standardizzate che permettessero lo sviluppo di software Bluetooth indipendente dalla piattaforma.

Quando si decise di creare questo set di API standardizzate per la tecnologia wireless Bluetooth, la scelta del linguaggio di programmazione da utilizzare ricadde su Java. Oltre ai benefici della portabilità, che permettono alle API Java di essere eseguite su hardware, sistemi operativi e classi di dispositivi diversi, Java permette inoltre un rapido sviluppo delle applicazioni, grazie al suo paradigma di programmazione orientato agli oggetti che ne garantisce un'astrazione ad alto livello, ed una comunità di sviluppatori già molto estesa.

Il Java Community Process introdusse le prime specifiche di API standardizzate per il Bluetooth nell'anno 2000. Queste specifiche conosciute come JSR-82 (Java Specification Request 82) o JABWT (Java Api for Bluetooth Wireless Technology) definiscono le basi per lo sviluppo di applicazioni Bluetooth e sono state pensate e progettate per poter essere utilizzate anche da sistemi di tipo CLDC (Connected Limited Device Configuration) che hanno limitate capacità di calcolo, di memoria e batteria. Gli sviluppatori possono così creare delle applicazioni Bluetooth indipendenti dall'hardware utilizzato e che quindi possono essere implementate anche su device diversi purché questi supportino JSR-82. Per poter utilizzare le JABWT un dispositivo deve disporre di uno stack che sia qualificato per almeno il Generic Access Profile, il Service Discovery Application Profile ed il Serial Port Profile e che permetta l'accesso al Service Discovery Protocol ed ai suoi livelli RFCOMM e L2CAP.

Molta attenzione è stata posta anche alla scalabilità, in modo da permettere a questo tipo di applicazioni di poter essere eseguite su qualsiasi tipo di

piattaforma Java 2 (J2ME, J2SE, J2EE) dotata del Generic Connection Framework (GCF).
Proprio per motivi di scalabilità queste nuove API non implementano tutti i livelli ed i profiles indicati dalle specifiche Bluetooth ma si limitano a ricoprirne solo una parte. Per esempio sono supportati:

- la comunicazione dati e non quella vocale;
- l'utilizzo dei protocolli;
- L2CAP (solo nella versione orientata alla connessione);
- RFCOMM;
- SDP;
- OBEX (OBject EXchange protocol)

I profiles supportati sono:

- Generic Access Profile (GAP);
- Server Discovery Access Profile (SDAP);
- Serial Port Profile (SPP);
- Generic Object Exchange Profile (GOEP)

Le funzionalità che mettono quindi a disposizione le JABWT sono:

- la ricerca di dispositivi e servizi;
- la registrazione dei servizi;
- la possibilità di stabilire connessioni di tipo RFCOMM, L2CAP e OBEX;
- la possibilità di eseguire queste operazioni in una modalità "sicura".

Ringraziamenti

Che dire... mi sono ritrovato a scrivere questa tesi a meno di due anni da quella per la laurea di primo grado. Chi l'avrebbe mai detto? Eppure è così, con la sola differenza che questa è molto più importante perchè segna la fine dei miei studi in questa università. Un'università che mi ha preso tanto, ma che tanto mi ha anche saputo regalare: soddifazioni e gioie, traguardi intermedi e obiettivi (finalmente raggiunti), ma soprattutto amici e allegria. Di tutte queste cose, le ultime due saranno sicuramente quelle che mai dimenticherò.

Prima di ricordare gli amici voglio però ringraziare il mio docente e relatore *Paolo Giorgini*, che mi ha seguito e consigliato in questa come nella precedente tesi e mi ha indirizzato verso un posto di lavoro che, mi auguro, possa diventare permanente.

Un doveroso grazie va anche a *Luca Debiasi*, che mi ha accolto e inserito in ArsLogica nel migliore dei modi e mi ha fatto sentire da subito a mio agio. Sono sicuro che con lui inizierà fin da subito non solo un ottimo rapporto di lavoro, ma anche di stima e di amicizia, come del resto è stato fino ad ora.

Ed ora gli amici. Un grazie a quelli che fin dai primi mesi del primo

anno (correva il lontano settembre 2000) sono sempre stati con me: in particolare, *Pise*, compagno fedele fino alla fine (anche se poi le nostre strade si sono leggermente divise a causa di piani di studio differenti); e *Gerry*, che accidenti a lui si è perso per strada al terzo anno ma che forse ora finalmente ha ritrovato il sentiero giusto. E poi tutta la banda al completo in doveroso ordine alfabetico: *Ava, Bishops, Carlo, Chiara, Conf, Dorlo, Elisa, Fabri, Fra, Gabry, Lazzaro, Marche, Micky, Pol, Roby, Sat, Sep, Silvia, Sly, Teo, Viruz* e infine *Zeus*.

Un grazie immenso ad *Elena*, la mia ragazza, che tra le infinite altre cose ha sempre capito i miei impegni e mi ha sempre lasciato tutto lo spazio ed il tempo di cui avevo bisogno, specie negli ultimi mesi, da quando viviamo insieme. D'ora in poi non avrò più la scusa dello studio e ti "trascurerò" un po' meno, promesso!

Ultimi, ma sicuramente primi nella lista, i miei genitori, che mi hanno sempre sostenuto specie economicamente e che hanno sempre creduto in me, interessandosi sempre (forse anche troppo mamma?) di come procedevano gli studi e chiedendomi certi particolari pur sapendo a priori di non capire nulla della risposta... ma l'importante non è quello, ed io ho sempre apprezzato. Grazie davvero!

Una dedica speciale va infine anche alla nonna *Ida*; sono 10 anni che dici "non so se ci sarò quando diventerai dottore": visto, non solo ci sei ma sei anche in formissima e l'ormai rituale canzoncina è dedicata a te!

Elenco delle figure

Bibliografia

[1] Il sito ufficiale per il Bluetooth – *http://www.bluetooth.org/.*

[2] JSR-82: Java Api per il Bluetooth – *http://www.jpc.org/en/jsr/detail?id=82.*

[3] Progetto BlueCove – *http://www.sourceforge.net/projects/bluecove/.*

[4] Progetto MIA – *http://www.uni-koblenz.de/∼bthomas/MIA_HTML.*

[5] JADE: Java Agent DEvelopment Framework – *http://jade.tilab.com/.*

[6] FIPA: Foundation for Intelligent Physical Agents – *http://www.fipa.org/.*

[7] JACK, an environment for building, running and integrating commercial-grade multi-agent systems – *http://www.agent-software.com/shared/products/.*

[8] The MadKit Project, a Multi-Agent Development Kit – *http://www.madkit.org/.*

[9] LEAP: Lightweight Extensible Agent Platform – *http://leap.crm-paris.com/.*

[10] Implicit Culture – *http://dit.unitn.it/∼implicit.*

[11] PortableCicero – *http://giove.cnuce.cnr.it/Cicero.html.*

[12] Y. Shoham. An Overview of Agent-Oriented Programming. *In: J. M. Bradshaw, ed. Software Agents. AAAI Press/The MIT Press, p. 271 - 290 (1997).*

[13] E. Blanzieri, P. Giorgini. From Collaborative Filtering to Implicit Culture. *In Proc. of the Workshop on Agents and Recommender Systems, in Agents2000, Bercellona (http://www.science.unitn.it/∼pgiorgio/ic), 2000.*

[14] A. Rakotonirainy, S. W. Loke, and A. Zaslavsky. Towards Multi-Agent Support for Open Mobile Virtual Communities. *In Proc. of the International Conference on Artificial Intelligence (IC-AI 2000) (Vol I), Las Vegas, Nevada, USA, pages 127-133, 2000.*

[15] E. Blanzieri, P. Giorgini, P. Massa, and S. Recla. Implicit Culture for Multi-Agent Interaction Support. *In CooplS '01: Proc. of the 9th International Conference on Cooperative Information Systems, pages 27-39, London, OK, 2001. Spingers-Verlag.*

[16] M. Wooldridge. An Introduction to Multiagent Systems. *Published in February 2002 by John Wiley & Sons (Chichester, England).*

[17] M. Bombara, D. Calì, and C. Santoro. Kore: a Multi-Agent System to Assist Museum Visitors. *In Proc. of the Workshop on Object and Agents (WOA2003), Cagliari, Italy, pages 175-178, 2003.*

[18] C. Carabelea and O. Boissier. Multi-Agent Platforms on Smart devices: Dream or Reality? *In Proc. of The Smart Object Conference (SOC03), Grenoble, France, pages 126-129, 2003.*

[19] H. Rehingold. Smart Mobs: The Next Social Revolution. Basic Books; Reprint edition (October, 2003) ISBN: 0738208612.

[20] L. Vasiu and Q.H. Mahmoud. Mobile Agents in Wireless Devices. *Computer, 37(2): 104-105, February 2004.*

[21] O. Bocur, P. Beaune, and O. Boissier. Representing Context in an Agent Architecture for Context-Based Decision Making. *In Proce. of the Workshop on Context Representation and Reasoning (CRR'05), Paris, France, 2005.*

[22] P. Baroni, A. Gerevini, P. Toninelli. MAgentA: Un Sistema Multi-Agente per la Gestione di Agende e Riunioni. *In Proc. of the WOA05, pagg. 127-135, Camerino, Italy, November 2005.*