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

Corso di Laurea triennale in Informatica

Elaborato finale

BlueSharing: sistema Bluetooth di condivisione file per dispositivi mobili.

Relatore:
Ing. Paolo Giorgini

Laureando:
Cristian Stenico

Anno Accademico 2006 - 2007

Indice generale

Introduzione

Nell'ultimo decennio la tecnologia ha puntato molto sulla mobilità, rendendola di fatto un'opportunità di sviluppo e di espansione al pari di quella cablata. Ora infatti i dispositivi mobili (soprattutto cellulari, ma anche computer palmari e smartphone) hanno acquisito la possibilità di eseguire molti programmi e applicazioni, abbandonando sempre più il concetto "un compito per un dispositivo" e avvicinandosi sempre più a piccoli computers. Sono stati così introdotti nel mercato dispositivi con capacità e features sempre maggiori, sfruttando anche e soprattutto l'espansione delle tecnologie wireless, l'efficienza sempre maggiore dei processori e il costo sempre minore dei componenti hardware.

Le prime applicazioni sviluppate per questi dispositivi erano dei porting di software già presente nei pc: visualizzatori di immagini e di video, editor di testi, semplici giochi, eccetera. L'avvento del Bluetooth, una versione semplificata del Wi-Fi che permette la comunicazione senza fili con un basso consumo di energia, ha rappresentato un'altra piccola rivoluzione. Mentre il Wi-Fi è più adatto a computer stand-alone, il Bluetooth si è diffuso rapidamente nei dispositivi mobili e al momento attuale è presente nella maggior parte dei cellulari, dei PDA e degli

smartphone, anche di fascia bassa. In attesa quindi che anche il Wi-Fi (che è più veloce e ha più raggio d'azione del Bluetooth) si diffonda così largamente nei dispositivi mobili, si possono sfruttare tutti i vantaggi che il Bluetooth offre per creare delle applicazioni pensate e sviluppate appositamente per questi dispositivi. Infatti sono già presenti alcune pionieristiche applicazioni che sfruttano internet ed il Wi-Fi (ad esempio applicazioni peer-to-peer, o applicazioni voip), ma a causa di alcune limitazioni (limitata memoria di massa nei dispositivi, bassa diffusione del wi-fi nei prodotti di fascia bassa) non possono essere ancora sfruttate appieno.

L'obiettivo di questa tesi è la progettazione e lo sviluppo di un'applicazione per dispositivi mobili che sfrutta la tecnologia Bluetooth per condividere e scambiare file tra più utenti. Per rendere più interessante e performante il suo utilizzo da parte dell'utente, ogni file condiviso è associato ad una o più parole chiave che faciliteranno così la ricerca all'interno del network, inoltre l'utente potrà sempre avere sotto controllo i file in condivisione e potrà visualizzarne il contenuto in ogni momento.

Il linguaggio utilizzato per sviluppare l'applicazione è java, più precisamente la versione micro edition di java, sviluppata dalla Sun Microsystems appositamente per dispositivi mobili, che garantisce un'ampia compatibilità indipendentemente dal modello o dalla marca del dispositivo, in quanto la java virtual machine è

installata di default in quasi tutti i cellulari e dispositivi mobili. Ovviamente il vantaggio di avere un'applicazione portabile porta anche degli inconvenienti, che verranno analizzati in seguito.

La tesi si articola in tre capitoli. Il primo capitolo riguarderà una panoramica sul filesharing e sul funzionamento e l'architettura del Bluetooth. Il secondo capitolo verterà la progettazione del software e la motivazione delle scelte strutturali. Nel terzo capitolo, infine, sarà presentata la parte implementativa del software e si descriverà com'è stato realizzato.

Capitolo 1

Tecnologie usate per sviluppare BlueSharing

In questo capitolo si parla delle tecnologie e dei concetti che stanno alla base dello sviluppo di BlueSharing.

1.1 Cos'è il file sharing

Il file sharing è la condivisione di file tra più utenti all'interno di una rete comune. Può avvenire attraverso una rete con struttura client-server oppure P2P (peer-to-peer[1]).

Il modello peer-to-peer è molto indicato in ambito wireless (*senza fili*) in quanto consente di assemblare una rete ad hoc velocemente, senza costringere gli utenti a configurazioni di routine. Le reti wireless sono in genere meno affidabili delle reti cablate perché sono soggette ai “capricci” delle onde elettromagnetiche, alle interferenze, alla durata delle batterie dei dispositivi. L'approccio P2P fornisce quindi un modello che è molto adatto alle vicissitudini dei componenti di una rete wireless (BlueSharing seguirà infatti questo modello).

Le reti di file sharing non vanno confuse con reti che costituiscono un filesystem

1 Peer-to-peer: rete di computer o qualsiasi rete informatica che non possiede client o server fissi, ma un numero di nodi equivalenti (peer, appunto) che fungono sia da server che da client verso altri nodi della rete.

distribuito[2][1].

1.1.1 Un po' di storia

Quando Napster raggiunse nel 2000, in poco più di un anno, i 60 milioni di utenti, il file sharing esplose come fenomeno di massa catturando l'attenzione del grande pubblico, dei media e di tutte le major.

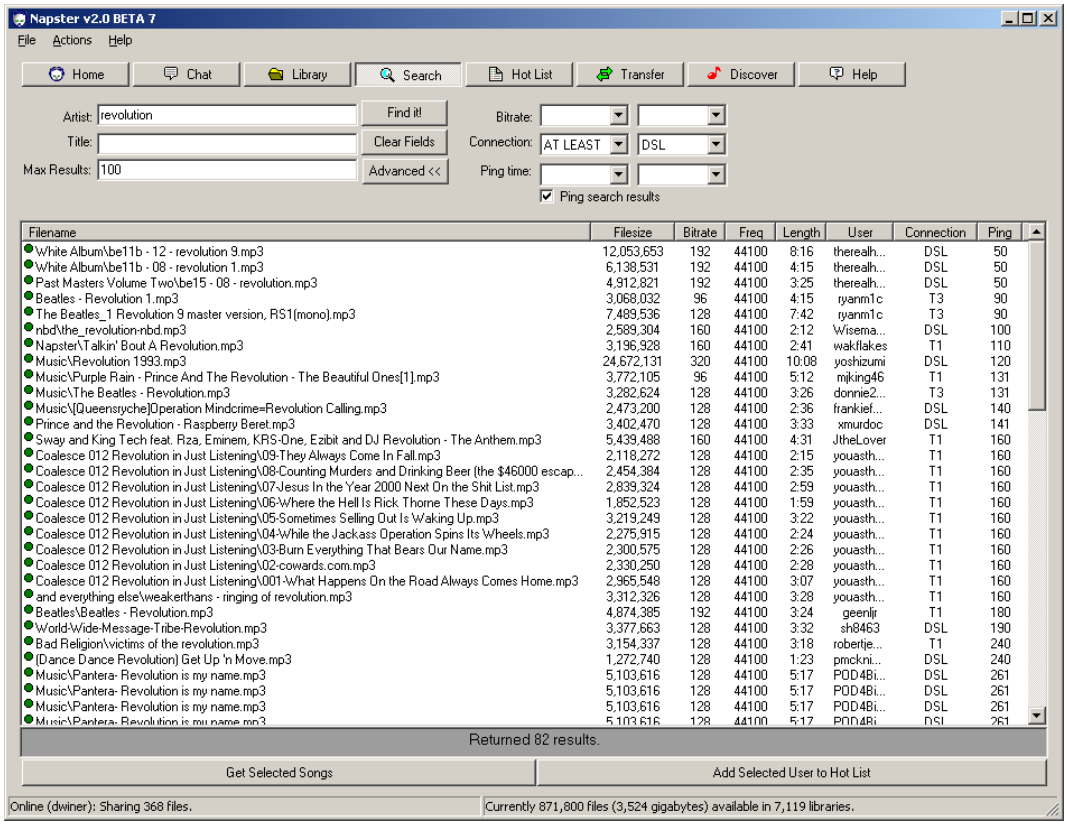

Figura 1: Screenshot di Napster nel 2001

In realtà Napster non fu il primo software di file sharing realizzato (di gran lunga preceduto dalle IRC), ma ebbe sicuramente il merito di catturare l'attenzione degli

2 Filesystem distribuito: particolare filesystem che permette la localizzazione di files e risorse in dispositivi di archiviazione distribuiti in una rete informatica.

utenti, grazie alla sua facilità d'uso ed intrinseca viralità.

Caratteristica tecnica di Napster era la struttura “centralizzata” della propria rete: in altri termini Shawn Fanning (inventore di Napster) aveva predisposto un'architettura basata su un server centrale di riferimento, una sorta di motore di ricerca, cui potevano collegarsi tutti i client. Ogni volta che un utente si collegava, riceveva le indicazioni dal server sugli utenti con cui poter scambiare file e stabilire a quel punto una connessione diretta (peer-to-peer, appunto).

Questa struttura, per quanto tecnicamente vincente, rese però Napster facilmente vulnerabile agli attacchi delle major, infatti fu sufficiente individuare e bloccare il server centrale per metterlo fuori uso. Dalla chiusura di Napster in poi, gli sviluppatori di software di file sharing hanno iniziato a creare sistemi sempre più delocalizzati, così da evitare che la chiusura di un unico server potesse compromettere il funzionamento dell'intera rete. Ecco nascere, quindi, reti ibride e completamente decentralizzate, caratterizzate da diversi software client in continua evoluzione (ad esempio Kazaa, WinMX, eMule). Questi software erano supportati da un'infrastruttura formata da molti server, rendendo di fatto impossibile l'interruzione della condivisione.

In risposta a questo, la RIAA (Recording Industry Assoxciation of America) e la MPAA (Motion Pictures Association of America), le associazioni delle case

discografiche americane, procedettero con cause e denunce a migliaia di utenti. I risultati di tali provvedimenti però non hanno dato gli esiti sperati: i verdetti emanati dagli organi giudiziari non erano quasi mai a sfavore degli utenti, inoltre gli sviluppatori introdussero nei loro software nuove funzionalità per tutelare gli utenti da nuove denunce[2].

Un esempio recente è il programma Mute, il quale:

- utilizza la crittografia per proteggere il contenuto dei file scambiati
- rende anonimo l'utente trasformando l'indirizzo IP in un IP virtuale
- non mette in connessione diretta i pc degli utenti che scambiano file tra loro (l'utente A, in base ad un algoritmo random, scarica da B passando ad esempio per C e D)

Ad oggi la situazione non è ancora ben definita, anche perché i software di file sharing di per sé non violano nessuna legge e nessun diritto d'autore e non possono quindi essere perseguiti, resta all'utente finale la responsabilità di usare tali programmi con coscienza e senza violare alcuna legge.

1.1.2 Filesharing su dispositivi mobili

Il panorama del file sharing localizzato al mondo mobile non è molto ricco, anche se è molto plausibile che lo possa diventare in un futuro prossimo.

Esiste un set di protocolli aperti chiamato JXTA ed è progettato espressamente per

Java e permette di creare facilmente applicazioni P2P, purtroppo però non è stato inizialmente pensato anche per J2ME; è nato successivamente il progetto JXME che mira a portare le funzionalità di JXTA su J2ME, ma non ha ancora raggiunto livelli usabili, in quanto i requisiti di molti protocolli di JXTA eccedono le risorse dei dispositivi mobili, per cui si sta sviluppando una forte ottimizzazione dei protocolli per renderli completamente disponibili anche su apparecchi mobili.

Stanno invece crescendo con un ritmo molto veloce le applicazioni di chat e Instant Messaging, esistono infatti Jimm, JimmyIM, Virca, WLIrc, Mobber, jmIrc (client per i più utilizzati protocolli di chat come Irc, MSN, ICQ, Jabber), che per funzionare necessitano però di una connessione internet.

Manca quindi all'appello un software che permetta di condividere e scambiare files svincolato dal requisito di una connessione ad internet[3].

1.2 Java 2 Micro Edition (J2ME)

J2ME è un ambiente di sviluppo Java fortemente ottimizzato ed è stato concepito per creare applicazioni in grado di essere eseguite su dispositivi mobili ed embedded.

Dato che esistono molti sistemi operativi fra la moltitudine di modelli in circolazione, è compito delle case produttrici implementare una JVM[3] per ogni

3 JVM: Java Virtual Machine, ovvero la macchina virtuale adibita all'interpretazione e

specifico dispositivo.

1.2.1 Configurazioni e profili

I dispositivi mobili possono avere differenti forme, features e funzionalità, ma spesso usano processori e quantità di memoria simili, quindi le configurazioni vengono create definendo gruppi che hanno caratteristiche di processore e memoria similari. Le caratteristiche che vengono prese in considerazione per le configurazioni sono:

- Features supportate del linguaggio di programmazione Java.
- Features supportate dalla JVM.
- Le librerie base e le API Java supportate.

Esistono due configurazioni standard per la piattaforma J2ME: Connected Device Configuration (CDC) e Connected Limited Device Configuration (CLDC). CDC è progettato per dispositivi potenti come sistemi di navigazione per auto, mentre CLDC è progettato per apparecchi meno potenti come cellulari o PDA. In questa tesi si prenderà in considerazione solo la configurazione CLDC, in quanto BlueSharing è stato progettato per cellulari e smartphone.

Un *profilo* definisce un set di API che offrono l'accesso alle capacità specifiche del dispositivo. MIDP (Mobile Information Device Profile) è un profilo usato con la configurazione CLDC e fornisce un set di API che includono classi per

all'esecuzione del codice java.

l'interfaccia utente, per la memorizzazione dei dati e per l'utilizzo dei sistemi di comunicazione.

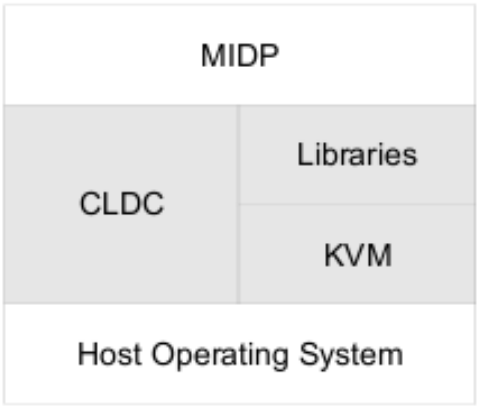

Figura 2: Posizione di CLDC e MIDP nell'architettura J2ME

Nell'immagine sopra si fa riferimento anche alla KVM (Kilo Virtual Machine), che altro non è che la JVM ottimizzata per J2ME, e viene chiamata in questo modo perché la sua grandezza è dell'ordine dei kilobyte[4].

1.2.2 JSR

Tutte le informazioni sulle tecnologie legate al mondo J2ME è possibile reperirle attraverso le JSR (Java Specification Request), che sono delle collezioni di API che si concentrano su una funzionalità specifica. In questo modo la KVM in ogni modello di dispositivo supporterà solo le JSR che riescono ad essere eseguite in quel determinato modello, ottimizzando quindi la quantità di memoria occupata. Per fare un esempio: un apparecchio che non dispone di funzionalità Bluetooth, nella sua JVM non ci sarà il supporto per la JSR-82, che è quella adibita alla

comunicazione Bluetooth, e così via. Le principali JSR sono:

- JSR-120: WMA (Wireless Messaging API), ossia sono le API che permettono di inviare e ricevere messaggi di testo (SMS), multimediali (MMS) o binari. Viene inoltre definito il concetto di Push Registry, ovvero un registro che permette di avviare un'applicazione grazie ad un SMS che viene ricevuto dal dispositivo.
- JSR-82: JABWT (Java Bluetooth Wireless Technology), sono le API che descrivono completamente le interazioni che è possibile fare da Java con la comunicazione Bluetooth. Queste API sono disponibili sia per J2ME, dove vengono implementate dai costruttori, sia per J2SE, con librerie che aderiscono allo standard.
- JSR-172: WSA (Web Services API), sono delle API che permettono di implementare un client che sfrutta un WebService. Alla base di questa API ci sono librerie per il parsing dell'XML e per l'invio di richieste stile RPC.
- JSR-75: FileConnection, ossia delle API che permettono l'interazione con il filesystem del dispositivo (ram o memory card esterne)e con le classiche informazioni che possiamo avere su un cellulare, come rubrica, todo, note. Questa JSR è fortemente sfruttata in BlueSharing, ma, essendo relativamente giovane, non tutti i modelli compatibili con J2ME la

supportano,

Esistono inoltre altre JSR per il render delle immagini, per l'utilizzo di connessioni GPS, per creare grafica 3D, per gestire database, effettuare chiamate ed altre ancora, ma essendo molto avanzate e molto giovani, i dispositivi che le supportano sono ancora relativamente pochi[5].

1.2.3 MIDlets

Le applicazioni MIDP sono chiamate MIDlets, e vengono rese disponibili in MIDlet suites.

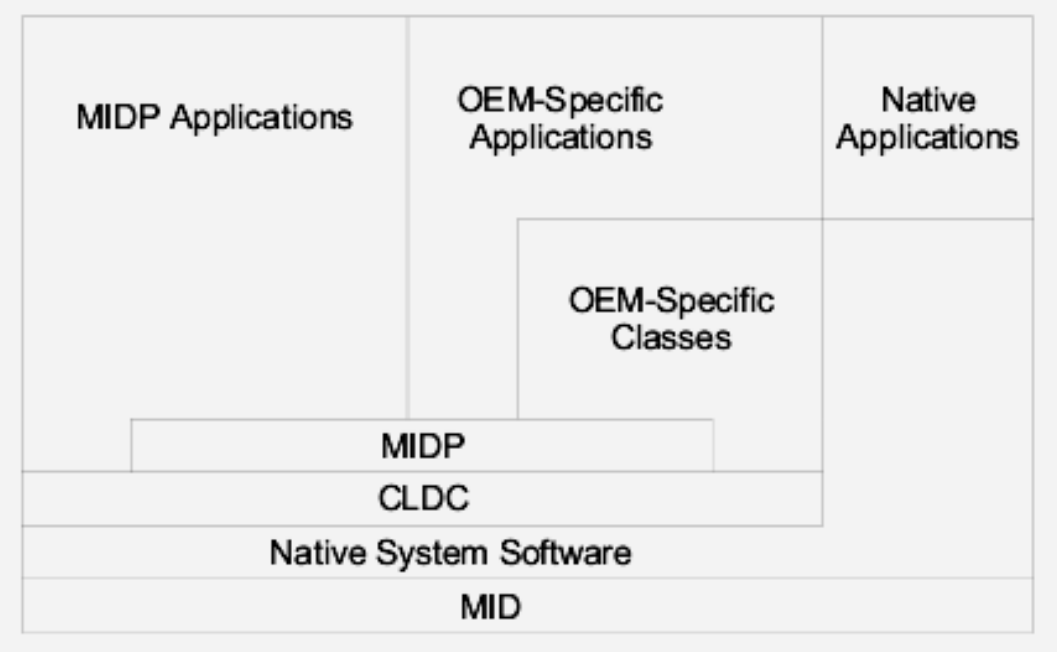

Figura 3: Architettura della MIDlet

Le applicazioni OEM-Specific si basano sulle classi OEM-Specific, che sono delle classi per le interfacce utente o l'accesso a funzionalità del device proprie delle varie case produttrici (ad esempio la vibrazione del dispositivo è gestita da

queste classi). Pertanto se vengono sfruttate queste classi, si rimarrà vincolati alla marca o addirittura al modello su cui l'applicazione è stata progettata, rendendo vano perciò il vantaggio della portabilità delle MIDlets.

MIDP definisce un modello di applicazione che lascia condividere le risorse limitate del device da MIDlets multiple. Il modello definisce:

- Cos'è una MIDlet
- Come viene impacchettata
- l'ambiente runtime adatto alla MIDlet
- il comportamento delle MIDlets per gestire le risorse del dispositivo
- l'impacchettamento delle MIDlets che formeranno una MIDlet suite

Una MIDlet suite consiste in due file: un file con estensione “jar” ed uno con estensione “jad”. Il file di archivio Java (jar) contiene le classi compilate in un formato compresso e preverificato. Un file Jar può contenere più di una MIDlet, permettendo la condivisione delle risorse da parte di tutte le MIDlets incluse. A causa delle restrizioni per la sicurezza una MIDlet può accedere alle risorse associate alla propria MIDlet suite.

Un file “jad” (Java Application Descriptor) è un file di testo che contiene le informazioni sulla MIDlet suite. Tutte le MIDlets devono essere nominate nel file jad, inoltre devono essere presenti la grandezza e l'URL del file jar, mentre il

numero rappresentante la versione della MIDlet suite è facoltativo. La presenza della grandezza del file jar è molto importante perché il MID (Mobile Information Device) può così determinare se c'è lo spazio necessario per installare la MIDlet suite oppure no.

Il MID scarica sempre prima il file jad per verificarne il contenuto. Se una MIDlet suite è già installata, riconoscerà la presenza di una nuova versione.

Se tutti i controlli vanno a buon fine, il MID segue l'URL per scaricare il file jar (attraverso un web server, ad esempio) e procedere con l'installazione. Nel file jad possono essere incluse altre informazioni, come il nome del produttore e dello sviluppatore, o eventuali certificati, che in alcuni modelli di dispositivi sono un requisito essenziale per essere installate o per accedere a particolari risorse.

In realtà attualmente è possibile installare una MIDlet anche senza la presenza di un file jad, avendo quindi solo il file jar, a patto che le politiche del dispositivo lo consentano, infatti molti modelli sono dotati di sistemi DRM[4] e quindi senza un appropriato certificato software (che in genere è a pagamento ed ha una durata limitata nel tempo) non è possibile installare nulla. Certi modelli, meno restrittivi, permettono di installare MIDlets ma ne limitano i privilegi qualora mancasse un certificato[6].

4 DRM: Digital Rights Management, ossia sono dei sistemi (hardware o software) con lo scopo di controllare l'autenticità e la legittimità dei contenuti. Esistono però seri dubbi sul beneficio che questi sistemi dovrebbero apportare.

1.3 Bluetooth

1.3.1 Cos'è Il Bluetooth

Il Bluetooth è una specifica industriale per reti personali senza fili (WPAN, Wireless Personal Area Network). Provvede un metodo standard, economico e sicuro per scambiare informazioni tra dispositivi diversi attraverso una frequenza radio sicura a corto raggio. Questi dispositivi possono essere ad esempio palmari, telefoni cellulari, personal computer, computer portatili, stampanti, fotocamere digitali, console per videogiochi.

La specifica Bluetooth è stata sviluppata da Ericsson e in seguito formalizzata dalla Bluetooth Special Interest Group (SIG), la cui costituzione è stata formalmente annunciata il 20 Maggio 1999 ed è un'associazione formata da Sony Ericsson, IBM, Intel, Toshiba, Nokia e altre società che si sono aggiunte come associate o come membri aggiunti.

Questo standard è stato progettato con l'obiettivo primario di ottenere bassi consumi, un corto raggio d'azione (da 10 a 100 metri) e un basso costo di produzione per i dispositivi compatibili. Ogni dispositivo Bluetooth è in grado di gestire la comunicazione con altri 7 dispositivi sebbene essendo un collegamento di tipo master-slave solo un dispositivo per volta può comunicare col server.

La regione di comunicazione tra device Bluetooth viene chiamata piconet ed ogni dispositivo può far parte di più d'una piconet (ad esempio in una comportarsi da master, in un'altra da slave) creando così una scatternet.

Il protocollo Bluetooth lavora nelle frequenze libere di 2,45 Ghz.

Per ridurre le interferenze il protocollo divide la banda in 79 canali e provvede a commutare tra i vari canali 1600 volte al secondo. La versione 1.1 e 1.2 del Bluetooth gestisce una velocità di trasferimento fino a 723,1 kbit/s, mentre la versione 2.0 gestisce una modalità ad alta velocità che consente fino a 10 Mbit/s, aumentando però la potenza assorbita. La nuova versione utilizza segnali più brevi e quindi riesce a dimezzare la potenza richiesta rispetto al Bluetooth 1.2 (a parità di traffico inviato).

Il Bluetooth non è uno standard comparabile al Wi-Fi, dato che quest'ultimo è un protocollo nato per fornire elevate velocità di trasmissione con raggio d'azione maggiore, a costo di una maggior potenza dissipata e di un hardware molto più costoso. Infatti il Bluetooth crea una Personal Area Network (PAN), mentre il Wi-Fi crea una Local Area Network (LAN). Il Bluetooth può essere paragonato al bus USB mentre il Wi-Fi può essere paragonato allo standard ethernet[7].

I dispositivi Bluetooth si dividono in 3 classi:

Classe	Potenza (mW)	Potenza (dBm)	Distanza (approssimativa)
Classe 1	100 mW	20 dBm	~100
Classe 2	2,5 mW	4 dBm	~10
Classe 3	1 mW	0 dBm	~1

Tabella 1: Classi dei dispositivi Bluetooth

1.3.2 La comunicazione Bluetooth

Ogni dispositivo dotato di funzionalità Bluetooth dispone di uno stack dei servizi, cioè delle informazioni relative ai servizi di cui è capace e dei protocolli supportati: altri apparecchi potranno far uso di queste informazioni per determinare la possibilità di interazione con i nodi della piconet. Questo è necessario perché 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, infatti ogni volta che un dispositivo accede per la prima volta ad una piconet, effettua una scansione di tutti i nodi presenti per capire come può interagire con essi. Per fare un esempio concreto, se un cellulare Bluetooth vuole trasferire un messaggio di testo a un PDA, potrà interrogare quest'ultimo per verificare che sia in grado di ricevere e leggere del testo in qualche modo. I servizi possono anche essere creati ad-hoc per programmi specifici: se si crea un'applicazione client-server che comunichi via Bluetooth è

bene creare un servizio che verrà quindi usato solo da questa applicazione, evitando l'interferenza con altre applicazioni o trasmissioni (BlueSharing ha un suo servizio dedicato, chiamato “BlueSharing”)[8].

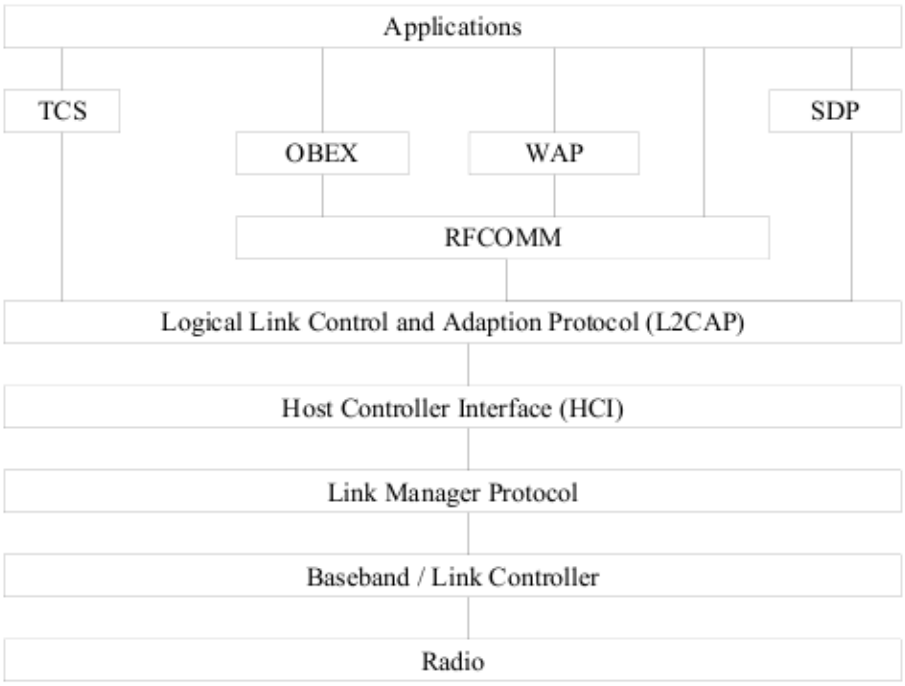

Figura 4: Stack dei protocolli Bluetooth

Livello	Descrizione
Applications	Provvede ad indicare come un'applicazione dovrebbe usare lo stack dei protocolli
Telephony Control System (TCS)	Servizio telefonico
Service Discovery Protocol (SDP)	Ricerca di servizi su dispositivi remoti
WAP e OBEX	Interfaccia tra i livelli più alti dei protocolli di comunicazione
RFCOMM	Fornisce un'interfaccia che simula una porta parallela RS-232
L2CAP	
HCI	Gestisce la comunicazione tra l'host ed il modulo Bluetooth
Link manager Protocol	Controlla e configura i collegamento con altri dispositivi
Baseband and Link Controller	Controlla i pacchetti e i collegamenti fisici
Radio	Modula e demodula i dati per la ricezione e la trasmissione

Tabella 2: Stack dei servizi Bluetooth

Ogni servizio possiede dei record che ne descrivono gli attributi:

Nome dell'attributo	ID dell'attributo	Tipo dell'attributo
ServiceRecordHandle	0x0000	32-bit unsigned integer
ServiceClassIDList	0x0001	Data Element Sequence (of UUIDs)
ServiceRecordState	0x0002	32-bit unsigned integer
ServiceID	0x0003	UUID
ProtocolDescriptorList	0x0004	Data Element Sequence (of UUIDs and protocol-specific parameters) or Data Element Alternative
BrowseGroupList	0x0005	Data Element Sequence (of UUIDs)
LanguageBaseAttributeIDList	0x0006	Data Element Sequence (of language parameters for supported languages)
ServiceInfoTimeToLive	0x0007	32-bit unsigned integer
ServiceAvailability	0x0008	8-bit unsigned integer
BluetoothProfileDesciptorList	0x0009	Data Element Sequence (of UUIDS)
DocumentationURL	0x000A	URL
ClientExecutableURL	0x000B	URL
IconURL	0x000C	URL

Tabella 3: Attributi dei servizi Bluetooth

1.3.3 Sicurezza

Nella comunicazione senza fili è molto importante avere un buon livello di sicurezza. Le specifiche Bluetooth definiscono un modello di sicurezza basato su

tre componenti: autenticazione, crittografia e autorizzazione.

L'autenticazione si divide in due fasi, chiamate *bonding* e *pairing*. La fase di bonding consiste nell'autenticare un device Bluetooth tramite una chiave di autenticazione condivisa. Se tale chiave non esiste, deve essere creata per completare il processo di bonding. La fase di pairing consiste nella creazione della chiave. Questa chiave è basata sull'input dell'utente e sull'indirizzo Bluetooth di uno dei due dispositivi. Appena digitato la chiave sul device locale, essa viene richiesta su quello remoto, per poi procedere con l'associazione dei due dispositivi, i quali, dopo aver terminato la fase di autenticazione, terranno in memoria la chiave condivisa per usi futuri.

La crittografia può essere utile nella trasmissione di informazioni o file dopo aver effettuato l'autenticazione. Prima che la crittografia abbia inizio, i due dispositivi devono accordarsi sul modo di crittografia usato e sulla dimensione della chiave crittografica. Ci sono tre tipi di crittografia utilizzabili: nessuna crittografia, crittografia dei pacchetti point-to point e broadcast e infine crittografia dei soli pacchetti point-to-point. Quando ci son solo due apparecchi connessi, la scelta indicata è ovviamente la terza, ovvero la crittografia dei pacchetti point-to-point. Il modello senza crittografia viene scelto son presenti dei dispositivi che non la supportano. Avviene perciò la contrattazione della dimensione della chiave di

crittografia, che può variare tra 8 e 128 bit.

L'autorizzazione serve per permettere a dispositivi remoti l'accesso a particolari servizi, e spesso richiede l'intervento dell'utente in risposta alla richiesta di accesso[9].

1.3.4 Perché il Bluetooth

La scelta del Bluetooth come protocollo di comunicazione per BlueSharing è dovuta al fatto che tra le tecniche di comunicazione wireless è la più diffusa e la meno costosa. Si stima infatti che gli apparecchi dotati di connettività Bluetooth venduti siano più di un miliardo. Inoltre la piattaforma J2ME utilizza le stesse API di comunicazione per vari protocolli (Bluetooth, Wi-Fi, Infra-Red), con l'unica differenza che consiste nello specificare il protocollo usato (identificato da un codice esadecimale) in fase di inizializzazione della connessione con un dispositivo remoto. Pertanto, nel caso di una futura diffusione della tecnologia Wi-Fi, basterà cambiare poche righe di codice in tutto il programma per permutare da un protocollo ad un altro.[10]

Capitolo 2

Progettazione del software BlueSharing

In questo capitolo presenteremo l'analisi e l'architettura di BlueSharing, ovvero tutte le operazioni precedenti all'implementazione del codice.

2.1 Possibili scenari

L'utilizzo del Bluetooth come tipo di comunicazione, pone di fatto l'esigenza di avere un network circoscritto in cui condividere e scambiare file.

Per questo il software non è sicuramente la scelta migliore nel caso si disponga di un accesso ad internet o ad una lan, mentre risulterebbe molto utile in situazioni opposte, come ad esempio in treno, dove l'unico requisito per ottenere una rete di file sharing è avere un dispositivo dotato di connettività Bluetooth.

BlueSharing è stato sviluppato proprio pensando a situazioni in cui il proprio dispositivo mobile è l'unico mezzo per comunicare ed interagire con altri apparecchi.

Supponiamo che ci sia uno studente che sta viaggiando in treno e decide di ascoltare della musica. In questo caso se è presente nei paraggi un altro utilizzatore di BlueSharing, lo studente può accedere ai suoi file musicali e scaricarli sul proprio dispositivo. Allo stesso modo, quei file scaricati in treno

saranno disponibili ad altri utenti in altre situazioni, come ad esempio una biblioteca, o un bar.

2.2 Funzionamento di BlueSharing

Figura 5: Schermata d'introduzione di BlueSharing. Disegnata con il software opensource GIMP

BlueSharing è composto principalmente da due parti: la prima consiste nella ricerca e nel download dei file, la seconda invece fa riferimento alla visualizzazione dei file già in possesso e in condivisione.

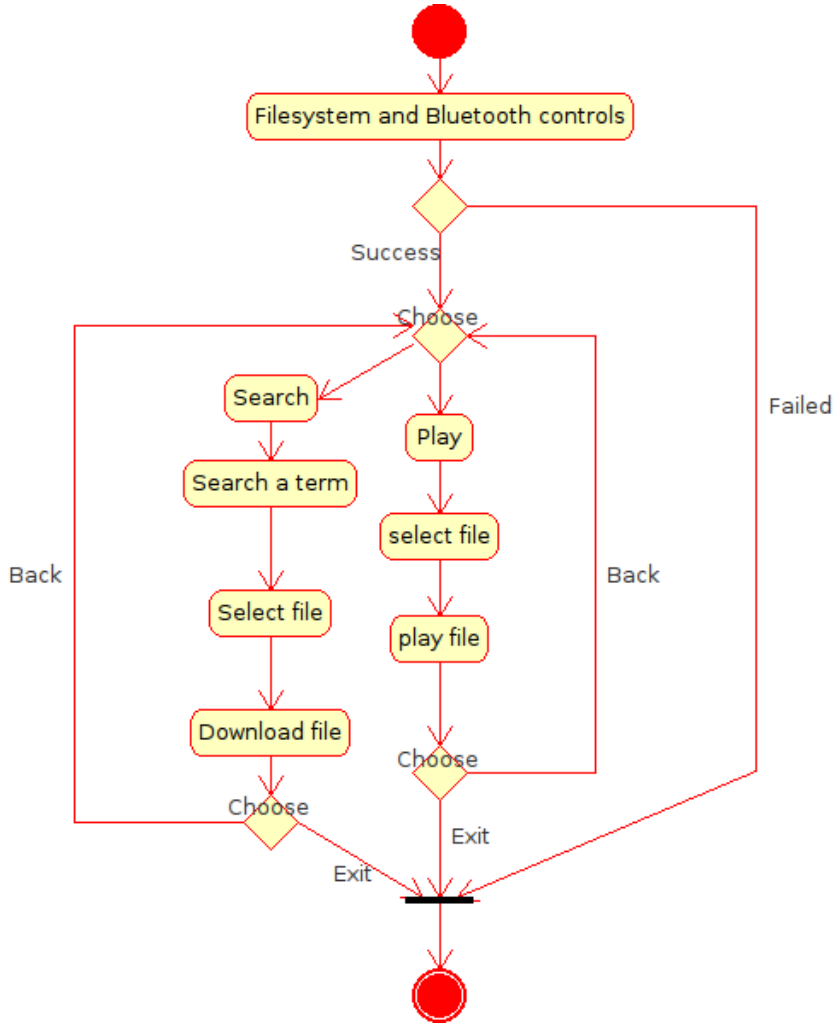

Figura 6: Activity Diagram

L'idea di fondo quindi è questa: all'avvio BlueSharing carica i dati sui file condivisi (qualora ce ne siano, ovviamente), controlla che esista la cartella adibita alla condivisione e controlla che il Bluetooth sia attivo sul dispositivo. Se una di queste operazioni non va a buon fine, il programma mostra una schermata d'errore e termina l'esecuzione. Se invece i controlli hanno un esito positivo, il programma fa partire il server, sempre attivo, parallelamente alle attività del client.

Quando tutte le operazioni di routine sono terminate, all'utente vengono proposte due scelte: cercare un file all'interno del network oppure visualizzare la lista dei file condivisi, per poterli eventualmente visualizzare. Se invece si rimane sulla schermata principale si potranno visionare le attività del server, che inserisce in un log tutte le richieste ricevute e i file inviati.

Dalla *form* di ricerca si può scegliere se effettuare la ricerca tramite un tag "base" (cioè uno tra quelli predefiniti: musica, foto, testo, varie) oppure eseguire una ricerca con una parola chiave personalizzata. Quando la ricerca è partita, il programma cerca il servizio apposito su tutti i dispositivi Bluetooth presenti nel raggio d'azione e quando lo trova chiede in risposta una lista di nomi dei file che trovano corrispondenza con la parola chiave (se presenti). Quando la ricerca è finita, l'utente ottiene quindi la lista di file che soddisfano la sua ricerca. A questo punto, per scaricare il file desiderato, è sufficiente selezionarlo. Mentre si sta effettuando il download, è presente una schermata che ne mostra all'utente l'avanzamento. Al termine del download, viene mostrata una schermata di notifica e si torna alla *form* centrale.

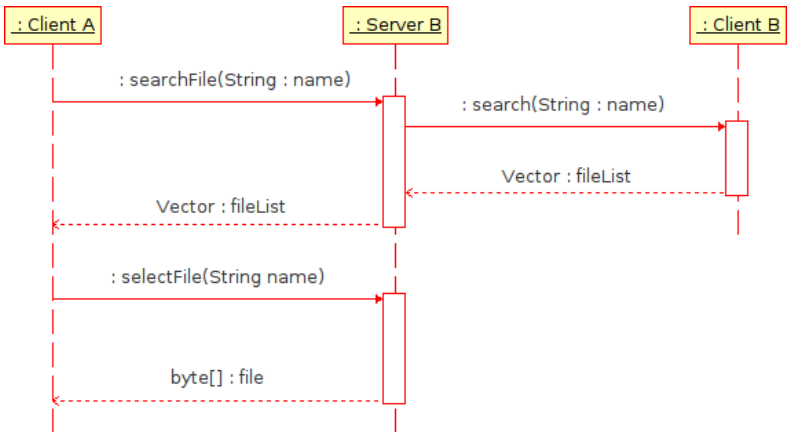

Figura 7: Sequence Diagram per il download di un file

Per la parte riguardante la visualizzazione dei propri file, dalla form principale, basta scegliere l'opzione “Play” anziché “Search”.

La schermata a cui fa riferimento tale comando è una lista contenente i nomi dei file in condivisione e un'icona che rappresenta il tipo del file:

-  per i file musicali (mp3)
-  per i file d'immagine (jpg, bmp, png, gif)
-  per i file di testo (txt)
-  per tutti gli altri tipi di file

Se si sceglie un tipo di file supportato, viene eseguito tenendo conto del tipo di file, ossia, in base al suo tipo si sceglie come aprirlo e cosa mostrare all'utente.

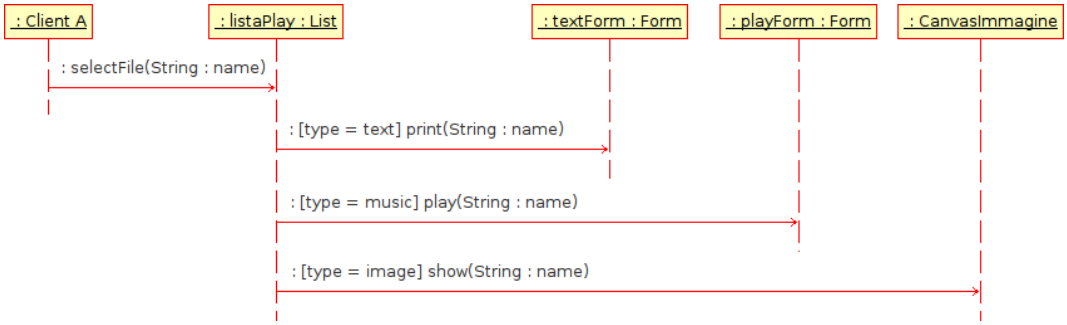

Figura 8: Sequence Diagram per la visualizzazione dei files

Nel caso di un file musicale, viene suonato, mostrando l'avanzamento della canzone, nel caso di un file immagine viene visualizzato a pieno schermo, mentre nel caso di un file di testo viene mostrato il suo contenuto in un'apposita *form.*

In tutti gli altri casi, viene mostrata una finestra di attenzione che avvisa che il tipo di file non è supportato.

Lo scopo di questa parte di BlueSharing non è quello di essere un player multimediale, bensì quello di dare l'opportunità di tenere sempre sotto controllo i file condivisi e di poterli visualizzare per controllare che i nuovi file non siano corrotti od incompleti.

2.3 Scelta degli strumenti di sviluppo

- Linguaggio di programmazione: Java, nella versione Micro Edition, la cui scelta è stata motivata nel capitolo 1. Le alternative sarebbero c++ e python, i quali, nonostante siano effettivamente più performanti e più potenti, hanno il difetto di essere vincolati al sistema operativo Symbian, a

differenza di java che è in grado di essere eseguito su qualunque sistema operativo che abbia una JVM (o una KVM) installata.

- Ambiente di sviluppo: Netbeans 5.5 assieme al *mobility pack* (è un plugin per Netbeans che lo abilita allo sviluppo di software in J2ME e che offre anche un comodo ed intuitivo designer per creare grafica le interfacce grafiche). L'alternativa sarebbe stata Eclipse, che però secondo la mia opinione è meno intuitivo e meno gradevole da usare.

2.4 Class Diagram

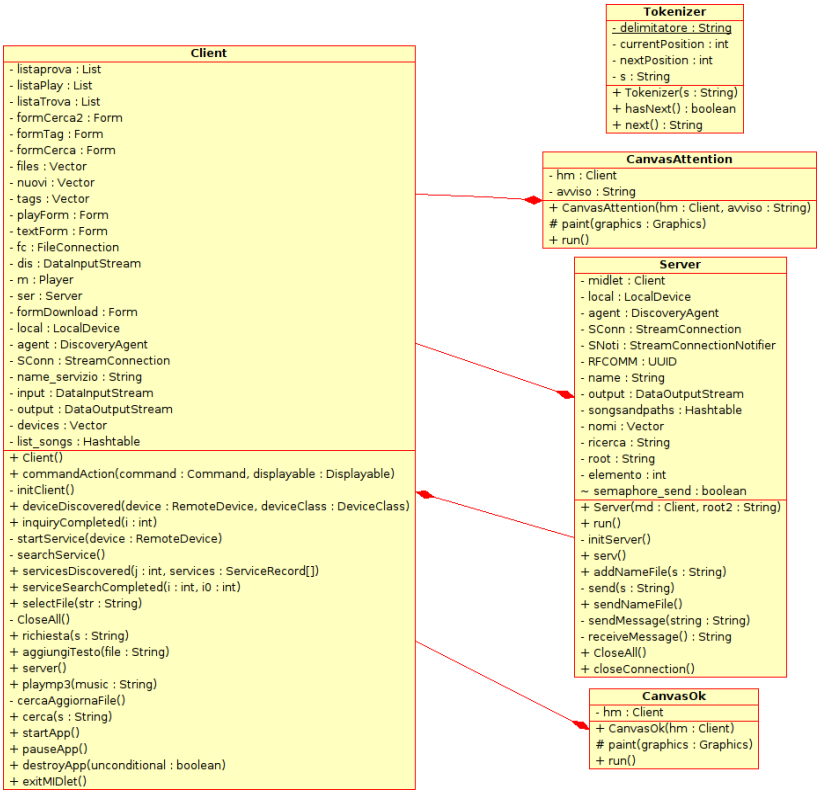

Figura 9: Class Diagram di BlueSharing

Questo è il Class Diagram che illustra i principali attributi e metodi di

BlueSharing. Ovviamente non sono presi in considerazione in questo schema gli oggetti grafici secondari (come i comandi, i Ticker, le Label, ecc...) in quanto non hanno un'importanza rilevante a livello di implementazione.

Le classi e le loro funzioni principali verranno illustrate nel dettaglio nel capitolo 3, comunque l'idea alla base di BlueSharing è di avere due classi principali, la classe Client e la classe Server. La prima serve per gestire i file condivisi, effettuare ricerche Bluetooth, chiedere file ai server remoti, mentre la seconda è un Thread sempre attivo e in attesa di qualche richiesta. Ci sono poi alcune piccole classi di supporto, come la classe Tokenizer, che serve a parsificare delle stringhe per ricavare le parole chiave associate ai file condivisi, e alcune classi Canvas che servono per mostrare messaggi all'utente.

Capitolo 3

Sviluppo del software BlueSharing

In questo capitolo analizzeremo BlueSharing dal punto di vista dell'implementazione e del testing. Verrano descritte le classi java che lo compongono, i test eseguiti per provarne la stabilità ed il funzionamento, e tutte le difficoltà affrontate in queste fasi.

3.1 Schema delle classi

3.1.1 La classe Client

Figura 10: Classe Client

La classe Client contiene tutti gli elementi grafici e i metodi che appartengono alla parte client.

In questa classe sono presenti metodi che fanno riferimento a quattro ambiti diversi:

- Salvare dati in locale tramite RecordStore
- Accesso al filesystem del dispositivo
- Ricerca e comunicazione Bluetooth
- Utilizzo di Player

Per salvare dati in locale nelle applicazioni J2ME, le API standard mettono a disposizione il Record Management System (RMS). L'RMS può essere organizzato in tabelle (RecordStore), identificate ciascuna da un proprio nome e costituiti da insiemi di record caratterizzati da un id e da un array di byte che ne rappresentano i dati. In BlueSharing il RecordStore (con nome “tags”) viene usato per memorizzare il nome dei file condivisi con le parole chiave associate.

Per scrivere dati in un RecordStore bisogna aprirlo con la chiamata

openRecordStore(“nome_del_RecordStore”, boolean createIfNotExist);

dopodiché si aggiungono i record con il metodo

addRecord(byte[] b, int start, int lenght);

si leggono i dati con

getRecord(int id, byte[] b, int offset);

e lo si chiude con il metodo

closeRecordStore();

Per cancellare un RecordStore si usa la funzione

deleteRecordStore();

In BlueSharing un record da salvare in un RecordStore è rappresentato da una stringa del tipo “NomeFile,Tag1,Tag2,Tag3,Tag4,” .

Figura 11: Ad ogni nuovo file condiviso, si chiede all'utente di identificarlo con una delle quattro opzioni

Il numero di tags associabili ad un file va da un minimo di uno ad un massimo di dieci. Il primo è obbligato e va scelto tra i quattro disponibili, come mostrato nella figura, mentre tutti gli altri sono facoltativi e a discrezione dell'utente.

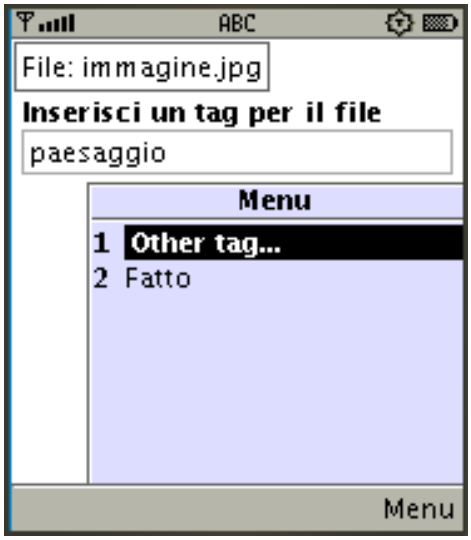

Figura 12: Inserimento dei tag aggiuntivi

Ad ogni inserimento di un nuovo tag, si può scegliere se terminare gli inserimenti o continuare, fino ad un massimo di dieci.

Ovviamente più un file ha delle parole chiave associate, più è facile ricercarlo nella rete.

Il RecordStore viene utilizzato solo all'apertura della MIDlet e alla chiusura.

All'apertura vengono caricati in un Vector i nomi dei file condivisi e in un altro Vector la lista di parole chiave dei suddetti file, poi il RecordStore viene cancellato. Alla chiusura della MIDlet (che può avvenire per esplicita scelta dell'utente o per qualche errore irreversibile) viene creato di nuovo il RecordStore caricando tutti i file con i tags relativi aggiornati. In questo caso quindi il RecordStore viene usato solo per memorizzare i dati anche se l'applicazione non è in esecuzione, mentre quando è in esecuzione vengono utilizzati oggetti più

maneggevoli come i Vector, limitando così anche ripetuti accessi al RecordStore.

Per quanto riguarda l'accesso al filesystem la classe principale a cui ho fatto riferimento è FileConnection, che fa parte delle API JSR-75. Tramite i suoi metodi è possibile aprire (in lettura e scrittura) e creare file e cartelle, anche se purtroppo queste API sono tra quelle meno supportate in generale dai telefoni cellulari.

Comunque, in BlueSharing vengono utilizzate in questo modo: come prima operazione effettua un controllo nella cartella “Incoming” che contiene i file condivisi. Se non esiste, la crea, mentre se esiste ottiene la lista di file contenuti in esse e quindi viene effettuato un controllo incrociato con i dati presenti nel RecordStore. Ovviamente se son presenti file nuovi viene chiesto all'utente di inserire i tag a cui devono essere associati, dopodiché vengono memorizzati assieme agli altri file. Se al contrario mancano file precedentemente presenti, ne viene cancellato il riferimento nel RecordStore.

Viene usato FileConnection anche quando si deve ricevere un file da remoto: si crea un file vuoto che verrà riempito con i byte ricevuti. Per la creazione di file e cartelle, è necessario aprire il file con permesso sia in lettura che in scrittura. Alla fine di ogni operazione su file ovviamente viene chiusa la connessione per liberare più risorse possibili.

Le classi che si interfacciano con la connessione Bluetooth sono altrettanto

utilizzate in BlueSharing, infatti vengono sfruttate le API per la ricerca dei dispositivi, per la ricerca dei servizi e per la connessione con dispositivi remoti.

Quando l'utente effettua un ricerca (tramite una parola chiave predefinita o personalizzata) viene chiamato il metodo *StartInquiry()* (ereditato dall'interfaccia *DiscoveryListener*) che provvede a ricercare tutti i dispositivi dotati di Bluetooth all'interno del suo raggio d'azione. Ad ogni apparecchio rilevato viene chiamata la funzione *deviceDiscovered(RemoteDevice d, DeviceClass dc)* dove *d* e *dc* sono rispettivamente un oggetto *RemoteDevice* che identifica il dispositivo trovato e la sua classe di appartenenza. In questa funzione l'oggetto *RemoteDevice* viene inserito in un Vector, che tornerà utile per la ricerca dei servizi Bluetooth. A ricerca terminata viene chiamata la funzione *inquiryCompleted(int i)* dalla quale si può far partire la ricerca dei servizi con la chiamata al metodo *serviceSearch(RemoteDevice rd),* che non appena trova tutti i servizi Bluetooth disponibili nel dispositivo esegue la funzione *servicesDiscovered(int j, ServiceRecord[] services)* nella quale si controlla se il dispositivo espone il servizio di BlueSharing, dopodiché viene eseguito *serviceSearchCompleted()* che indica che la ricerca dei servizi del dispositivo è terminata, inoltre comunica gli eventuali errori sorti nella ricerca. Se il servizio “BlueSharing” è stato trovato si apre la connessione verso il dispositivo per richiedere e ricevere la lista di file che

trovano una corrispondenza con la chiave di ricerca.

Tutti i nomi dei file trovati nella ricerca vengono inseriti in una HashMap assieme all'url del dispositivo che possiede tale file. In questo modo quando l'utente ha la lista dei file disponibile e sceglie di scaricarne uno, si userà l'url presente nella HashMap associato al nome del file scelto per connettersi al dispositivo remoto e procedere al download.

La sezione Play, infine permette di aprire e mostrare il contenuto di alcuni tipi di file in condivisione.

Figura 13: Lista dei file condivisi

Per ora sono supportati i file di testo (txt), i file immagine (jpg, bmp, png) e i file musicali (mp3). I file di testo vengono visualizzati in un'apposita *Form,* i file immagine vengono visualizzati in una *Canvas* fullscreen con dimensioni adeguate alla grandezza dello schermo, mentre i file musicali vengono caricati da un istanza della classe Player mentre sullo schermo appare una *Form* apposita contenente il

nome del file in esecuzione ed un *gauge,* il cui valore viene costantemente aggiornato da un Thread dedicato, e che indica il tempo trascorso nell'esecuzione.

Figura 14: Visualizzazione di un'immagine

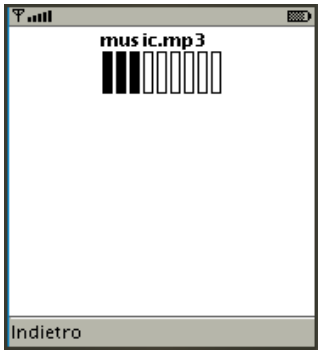

Figura 15: Esecuzione di un file musicale

Figura 16: Visualizzazione di un file di testo

Nel caso si selezioni un file con un'estensione non supportata, verrà mostrata la seguente *Canvas* di attenzione:

Figura 17: Canvas di attenzione

3.1.2 La classe Server

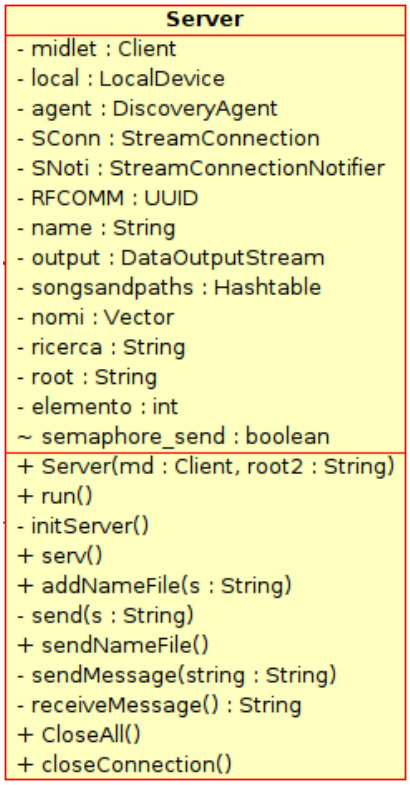

Figura 18: Classe Server

La classe Server è un Thread sempre attivo e in attesa di richieste da qualche client. Il suo compito è quello di creare e rendere visibile il servizio Bluetooth "BlueSharing", di inviare le liste di file durante le ricerche di un client e ovviamente di inviare i file scelti.

Per creare il servizio Bluetooth, basta creare un URL da associare al dispositivo locale per poi aprire la connessione e rimanere in ascolto su quell'URL.

Un URL Bluetooth è una stringa con la seguente formattazione:

btspp://localhost:RFCOMM_url;name=BlueSharing;authorize=true

dove al posto di RFCOMM_url va inserito un UUID (*Universally Unique*

Identifiers) per il protocollo RFCOMM e come nome si può inserire qualunque nome si voglia dare al servizio che si rende disponibile.

Appena giunge una richiesta da un client, il server valuta il tipo di richiesta e si comporta di conseguenza. Se la richiesta è una ricerca, allora il server attende in input la chiave di ricerca, effettua una ricerca tra i file condivisi e i loro tags ed eventualmente invia tutti i nomi dei file che corrispondono. Se invece la richiesta è un file, il server apre in lettura tale file, crea un buffer per la comunicazione , attraverso il quale invia il file al client. Ogni volta che termina la comunicazione con un client, la connessione viene terminata e riavviata in attesa di nuove richieste da servire.

3.1.3 La classe Tokenizer

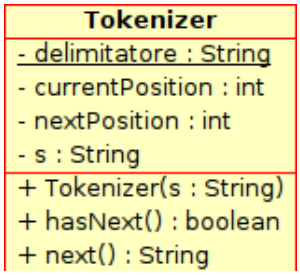

Figura 19: Classe Tokenizer

La classe Tokenizer è una classe presente nelle API di J2EE ma purtroppo non in quelle di J2ME. Questa classe serve a dividere una stringa in sottostringhe usando un carattere come delimitatore tra esse (chiamate “token”).

Ad esempio, se abbiamo la stringa “ciao,come,va,” il Tokenizer la dividerà in “ciao”, “come” e “va”, se ovviamente assumiamo che il carattere “,” sia il delimitatore.

Dato che in BlueSharing nel RecordStore vengono salvate stringhe contenenti sia i nomi dei file condivisi che i loro tag associati, è stato necessario implementare una versione di Tokenizer per J2ME che soddisfasse i requisiti richiesti in BlueSharing (infatti in questa implementazione accetta come solo parametro la stringa da dividere e assume il carattere “,” come delimitatore, a differenza della versione per J2EE, che accetta anche il delimitatore).

3.1.4 Le Canvas

La MIDlet contiene alcune classi che estendono la classe Canvas e sono usate per comunicare un evento (ad esempio un errore, o una notifica di un download terminato) all'utente.

La classe Canvas è una libreria che consente di creare un'interfaccia grafica a basso livello, dando come possibilità il disegno di rettangoli, ellissi, importazione di immagini e scrittura di testi. Per contro, avere a disposizione solo strumenti grafici primitivi rende abbastanza difficoltoso creare interfacce complesse e gradevoli da usare. Comunque, in BlueSharing non verranno usate per rimpiazzare completamente la grafica del programma, ma solo per comunicare determinati

eventi all'utente.

Il motivo della creazione di queste schermate di notifica è innanzitutto avere uno stile grafico coerente e svincolato dal sistema operativo sottostante (infatti ogni sistema operativo gestisce a modo suo le schermate di Alert), e dare un look più eye-candy a tali schermate. Inoltre in J2ME quando si visualizza una Canvas l'input dell'utente non viene più catturato dal metodo

commandAction(Command c, Displayable d)

ma il suo compito viene rimpiazzato dalla funzione *keypressed(int keyCode),* la quale accetta un intero che identifica un tasto sulla tastiera e quindi consente di gestire l'input al meglio, evitando pressioni di tasti non previste. Ad esempio: una schermata di Alert può essere resa visibile per un tempo predeterminato (per la durata della ricerca di dispositivi, ad esempio) ma la pressione di un qualunque tasto causa la sua cancellazione dallo schermo, quindi premendo altri tasti si può inavvertitamente causare un conflitto con l'operazione in atto. Con le Canvas invece si cattura l'input solo quando è necessario farlo.

Ci sono quattro colori usati per le varie Canvas, che servono a dare una metafora cromatica al messaggio che contengono:

- Blu: ricerca Bluetooth
- Rosso: Errore

- Giallo: Attenzione
- Verde: Notifica di download completato

Tutte le Canvas condividono lo stesso schema:

- Colore di sfondo
- Descrizione dell'ambito
- Descrizione specifica dell'operazione
- Icona identificativa

Avendo tutte le Canvas i testi personalizzabili possono essere usate in altri progetti liberamente importando la classi.

Di seguito quattro esempi delle quattro diverse tipologie di Canvas usate.

Figura 20: Canvas blu per la ricerca Bluetooth

Figura 21: Canvas rossa per i messaggi di errore

Figura 22: Canvas gialla per i messaggi di attenzione

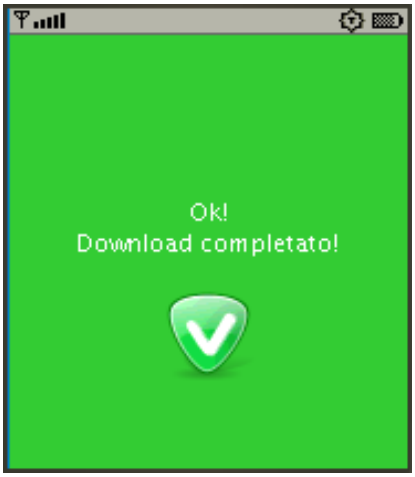

Figura 23: Canvas verde per il completamento di un download

Ci sono tre classi Canvas in BlueSharing:

1. un tipo estende la classe Thread, ed è associato agli errori. E' necessario che sia un Thread perché deve rimanere visibile sullo schermo per 5 secondi, per poi far terminare il programma.
2. anche il secondo tipo estende la classe Thread, ma dato che è associato alle avvertenze e alle notifiche di download, anziché terminare l'esecuzione della MIDlet, mostra la Form principale.
3. il terzo tipo non è un Thread, ed è usato solamente durante la ricerca Bluetooth. Questo vuol dire che la Canvas rimarrà visibile finché un evento esterno non interviene (in questo caso il metodo chiamato a fine ricerca).

3.2 Testing

Per poter eseguire correttamente BlueSharing il dispositivo candidato deve soddisfare alcuni requisiti:

- deve permettere l'installazione di MIDlets sprovviste di certificati software.
- deve essere compatibile con le JSR-82 e JSR-75 ossia con le API per la gestione del Bluetooth e del filesystem (illustrate nel capitolo 1).
- certi dispositivi, nonostante siano provvisti delle due caratteristiche sopracitate, non consentono la creazione di cartelle e certi altri lo permettono solo nelle schede di memoria e non nella memoria principale.

Per cui le restrizioni sono purtroppo molte, anche se in futuro dovrebbero lentamente sparire.

3.2.1 Modelli usati per il testing

Nonostante con l'ambiente di sviluppo per J2ME sia disponibile un emulatore software della KVM, non è stato possibile testare BlueSharing col suddetto emulatore, in quanto non è possibile effettuare l'accesso al filesystem, per cui il testing è stato effettuato esclusivamente sui dispositivi fisici in dotazione.

I modelli presi in considerazione sono: NOKIA 6630, NOKIA 6600 e NOKIA 6230i e su tutti e 3 i modelli il programma è funzionante.

Figura 24: Nokia 6630

Figura 25: Nokia 6600

Figura 26: Nokia 6230i

E' stato fatto un tentativo anche su un Sony Ericsson Z530i ma purtroppo questo modello non permette l'accesso al filesystem della memoria principale (e non è dotato di schede di memoria aggiuntive), per cui il programma si è installato correttamente ma restituisce un'eccezione non appena lo si esegue.

Dai test eseguiti sui sopracitati cellulari, sono sorti alcuni problemi, soprattutto nei modelli 6630. Sono purtroppo degli inconvenienti legati al modello stesso (in quanto ogni modello decide quanta memoria e quante risorse dedicare alla KVM) e in alcuni casi al sistema operativo ospitante:

- nel modello NOKIA 6630 se si ascolta più di 4 (a volte 3) volte dei file musicali si ottiene un errore del sistema operativo Symbian (“out of memory error”), in quanto l'allocazione del file musicale rimane anche

quando l'esecuzione è finita. Nonostante i provvedimenti presi contro questo errore (chiamare la Garbage Collector, porre a “null” tutti gli oggetti interessati all'esecuzione del file musicale) il problema persiste, mentre negli altri modelli non è stato riscontrato il problema, sebbene dotati di minore memoria.

- In alcuni casi la trasmissione di un file da un dispositivo all'altro può essere estremamente lenta, in quanto il sistema operativo Symbian non permette di avere dei buffer di trasmissione di grandezza adeguata alla velocità del Bluetooth. Ad esempio il modello 6630 riesce a trasmettere solo 500 byte ogni ciclo.
- Nei cellulari con sistema operativo Symbian, a meno di possedere un certificato, ad ogni accesso al filesystem e ad ogni connessione Bluetooth in entrata compare una finestra di dialogo che chiede all'utente di permettere all'applicazione di accedere alle risorse. Questo è abbastanza fastidioso soprattutto all'apertura del programma, quando cioè deve accedere alla cartella dedicata ai file condivisi e controllarne il contenuto, in quanto compaiono tre finestre di dialogo ai quali bisogna rispondere positivamente prima di poter usare il software.
- Ci sono svariati problemi legati strettamente al sistema operativo ospitante

che riguardano la visualizzazione grafica: se si passa da un oggetto Displayable contenente un Ticker ad una finestra Canvas, spesso il Ticker rimane visibile, come mostrato in figura.

Figura 27: Il Ticker (in alto a destra) rimane visibile anche nella Canvas

Un altro errore accade quando si vuole mostrare una nuova finestra Displayable con un codice di questo tipo:

```
getDisplay().setCurrent(get_form1());  //mostra la Form form1
selectFile(listaTrova.getString(listaTrova.getSelectedIndex()));
//scarica dal server il file selezionato
```

in questo caso la Form form1 non viene mostrata finché il file non viene scaricato completamente, nonostante compaia prima il comando di mostrare la Form. Anche questo problema si verifica solo nei modelli con sistema operativo Symbian.

- Purtroppo non è possibile attivare il Bluetooth da J2ME, perciò per far funzionare correttamente BlueSharing bisogna assicurarsi che sia attivo (in caso contrario, verrà mostrata a video una Canvas errore e il programma terminerà l'esecuzione).

Conclusioni e sviluppi futuri

L'obiettivo di questo elaborato era quello di sviluppare un software portabile di file sharing attraverso connessioni Bluetooth tra dispositivi mobili. Il software, nonostante qualche difficoltà relative alle piattaforme usate (hardware e software) risulta funzionante e stabile, almeno in alcuni modelli di dispositivi presi in considerazione per la fase di testing, ed ha tutte le caratteristiche di un (seppur rudimentale) software peer-to-peer. Non è attualmente possibile comunque garantire il suo funzionamento su tutti gli apparecchi che hanno un teorica compatibilità con i requisiti di BlueSharing, in quanto andrebbe testato su ognuno di loro per essere certi del suo funzionamento, inoltre ogni casa di produzione può applicare delle restrizioni sull'utilizzo delle risorse a propria discrezione.

In futuro si auspica che le enormi differenze che attualmente ci sono tra i vari dispositivi vengano totalmente cancellate, permettendo così agli sviluppatori di implementare software sicuramente più portabili. Per quanto riguarda BlueSharing, pur essendo già usabile e senza bug riscontrati in fase di testing, può beneficiare in futuro di nuove funzionalità e di miglioramenti. Ad esempio si potrebbero implementare delle classi che permettono di visualizzare file pfd o filmati o altri tipi di file ancora.

Un'altra modifica futura può essere l'affiancare alla comunicazione Bluetooth,

quella Wi-fi, per espandere le possibilità di condivisione.

Ci sono molte altre migliorie possibili, che possono avvicinare BlueSharing ai moderni sistemi di peer-to-peer, come ad esempio l'implementazione del download multiplo, cioè se un file che si vuol scaricare è presente su più dispositivi remoti, si può scaricarlo contemporaneamente da tutte le fonti disponibili, oppure permettere di riprendere a scaricare un file non completato.

Infine, un particolare forse marginale ma sicuramente importante, sarebbe la riscrittura dell'interfaccia grafica usando completamente le Canvas, svincolandosi così dall'interpretazione che ha ogni dispositivo delle API grafiche di J2ME. Le librerie grafiche sviluppate da Paolo Rallo per il client del progetto Andiamo potrebbero essere utilizzate per questo scopo.

Bibliografia

[1]: David Barkai, *Peer-to-peer computing (Technologies for Sharing and Collaborating on the Net)*, Intel Pr, 2002

[2]: Breve storia del file sharing

http://www.p2pforum.it/forum/archive/index.php/t-10714.html

[3]: Wallace Wang, *File sharing: guida non autorizzata al download*, Apogeo, 2006

[4]: Sito web della Sun Microsystem dedicato a J2ME
http://java.sun.com/javame/index.jsp

[5]: Sito degli sviluppatori Sun dedicato alle specifiche delle JSR
http://java.sun.com/javame/reference/apis.jsp

[6]: Pagine web della Sun per gli sviluppatori

http://developers.sun.com/

[7]: Sito ufficiale della tecnologia Bluetooth
http://www.bluetooth.com

[8]: B. Hopkings and R. Antony, *Bluetooth for Java*, First Edition, Apress, 2003

[9]: D. Gratton, *Bluetooth Profiles, The Definitive Guide*, First Edition, Prentice Hall, 2003

[10]: Kumar et. al., *Bluetooth Application Programming with the Java APIs*, First Edition, Morgan Kaufmann, 2004.

[11]: J. Knudsen, *Wireless Java: Developing with J2ME*, Second Edition, Apress, 2003

[12]: Dinesh Verma, *Legitimate applications of peer-to-peer networks*, Wiley-IEEE, 2004

[13]: Forum ufficiale Nokia

http://www.forum.nokia.com