Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Martin Kontsek

Mobilní komunikační software

Ústav formální a aplikované lingvistiky

Vedoucí bakalářskej práce: Mgr. Pavel Machek
Studijní program: Informatika, programování

2007

Ďakujem svojmu vedúcemu ročníkového projektu a bakalárskej práce za podporu, racionálne argumenty a dobré smerovanie práce.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne Martin Kontsek

Obsah

Název práce: Mobilní komunikační software
Autor: Martin Kontsek
Katedra (ústav): Ústav formální a aplikované lingvistiky
Vedoucí bakalářské práce: Mgr. Pavel Machek
e-mail vedoucího: pavel@ucw.cz

Abstrakt:
Cieľom práce je vytvorenie softvéru pre mobilné telefóny, ktorý bude vedieť urobiť sieť z okolitých mobilných telefónov využívajúci tento softvér cez protokol Bluetooth. Bude možné posielať po tejto sieti vlastné vytvorené média, a to text, audio, fotky a video. Média bude možné posielať aj cez protokol Jabber pomocou Uuencode/Uudecode, ktorý bude implementovaný v už existujúcom programu \"Mobber\". Program bude implementovaný v Java 2 Micro Edition a bude nadstavbou nad spomenutým programom "Mobber".

Klíčové slová: J2ME, Bluetooth, Jabber, komunikátor, mobilný telefón

Title: Communication software for mobile phones
Author: Martin Kontsek
Department: Institute of Formal and Applied Linguistics
Supervisor: Mgr. Pavel Machek
Supervisor's e-mail address: pavel@ucw.cz

Abstract:
The goal of this project is to write a program for mobile phones that will be able to create a network out of surrounding mobile phone by using the Bluetooth Protocol. It will be able to send media like text, audio, images and video (for those phones that support it). The program will also be capable of sending these media through the Jabber Protocol by using Uuencode/Uudecode. The program will be written in Java 2 Micro Edition a will be an add-on to an existing program "Mobber", in which the Jabber Protocol is already implemented.

Keywords: J2ME, Bluetooth, Jabber, communicator, mobile phone

Kapitola 1

Úvod

1.1 Mobilné telefóny a nástup „Bluetooth“

Napriek krátkej existencie sa mobilné telefóny stali pre mnoho ľudí nutnosťou. Sú niečím, bez čoho sa mnohí nezaobídu. K tomuto nezvyčajnému záujmu zo strany verejnosti sa výrobcovia mobilných telefónov snažili postupne pridávať nové vymoženosti. Okrem samotného telefonovania sa dalo aj posielať krátke správy, známe ako SMS. S rozširovaním pamäťovej kapacity prišla možnosť nahrávať či prehrávať zvuk. Ďalšou novinkou bola farebná obrazovka, ktorá vydláždila cestu pre dnes populárne zabudované kamery, čo umožnili fotenie, nahrávanie a prehrávanie videa. Prístup na Internet pomocou WAP dal, aj keď obmedzenú, možnosť prezerať obsah Internetu cez mobilný telefón. Vzhľadom na to, že mobilné telefóny sú v podstate „vysielačky“ a prístup na Internet bol a aj je spoplatňovaný, ďalším rozumným krokom bolo nejakým spôsobom umožniť priame spojenie dvoch telefónov na kratšiu vzdialenosť.

Tento krok ako prvá urobila švédska firma Ericsson, ktorá hľadala spôsob ako vytvoriť nizkonákladovú, energeticky nenáročnú rádiovú technológiu pre spojenie mobilného telefónu so svojim príslušenstvom. V roku 1998 spolu s ďalšími firmami vytvorili „ Bluetooth Special Interest Group“, teda skupinu, ktorá sa zaoberá rozvojom tejto technológie, ktorej dali názov „Bluetooth“ podľa dánskeho kráľa Haralda Blatanda[1]. Vyše 600 miliónov[2] predaných zariadení s „Bluetooth“ poukazuje na veľkú popularitu tejto technológie. „Bluetooth“ sa používa nie len v mobilných telefónoch, ale aj v PDA či v PC napr. na bezdrôtové spojenie s klávesnicou, myšou, tlačiarňou alebo aj mobilným telefónom. Možností je naozaj veľa.

1.2 Mobilné telefóny a Java 2 Micro Edition

Aplikácie pre mobilné telefóny (zvané MIDlety) sa programujú v platforme Java 2 Micro Edition (v skratke J2ME). V podstate je to jazyk Java s niekoľkými obmedzeniami, ktoré boli implementované práve kvôli pamäťovému a výkonnostnému hendikepu mobilných telefónov. Taktiež sú tu pridané triedy, ktoré sa využívajú na vytváranie a úpravu vzhľadu MIDletu.

Na spúšťanie programov napísaných v Jave sa používa Java Virtual Machine, ktorá prekladá Java bytecode (teda *.class súbory) do formy zrozumiteľnej pre architektúru, na ktorej sa program spúšťa. Kvôli spomínaným obmedzeniam bol pre

[1] “Blatand” v preklade znamená „modrý zub“, teda po anglicky „Bluetooth“

[2] http://www.bluetooth.com/Bluetooth/Press/News/Déjà_600_millions_de_terminaux_bluetooth_en_circulation.htm

tieto zariadenia[3] vyvinutý K[4] Virtual Machine. KVM je odvodené z JVM a je napísaný v jazyku C.

Na rozširovanie J2ME slúžia Java Specification Requests (JSR), čo sú špecifikácie daného rozšírenia. Ako príklad uvediem dve dôležité JSR, ktoré využívam vo svojej práci:

- *JSR-82:* Java API pre „Bluetooth“ – sada tried pre prácu s „Bluetooth“
- *JSR-135:* Mobile Media API – sada tried pre prácu s grafickým rozhraním

Každý výrobca mobilných telefónov sa sám rozhoduje, aké JSR implementuje do svojho výrobku. Avšak samotnou implementáciou sa nezaručuje, že bude zariadenie schopné využiť celú škálu možností, čo dané JSR poskytuje. Toto bolo zdrojom mnohých problémov, s ktorými som sa v priebehu programovania mojej aplikácie stretol.

1.2 Jabber

Ľudia odjakživa radi komunikujú a rozvíjajú sociálne vzťahy medzi sebou. Podľa niektorých vedcov je to vraj jeden z hlavných dôvodov prečo druh Homo Sapiens, na rozdiel od Neandertálca, prežil. V dnešnej dobe technológia umožňuje ľudom komunikovať na veľké vzdialenosti. S tým je však spojená aj otázka ceny prenosu informácii. Hlas sa prenáša ťažšie (a hlavne drahšie) ako text, preto si ľudia zvykli na písomnú formu komunikácie, a to čoraz častejšie pomocou IM[5] programov. Tieto programy spracovávajú a prenášajú text v rôznych formátoch zvaných protokoloch. Jedným z takýchto protokolov, resp. súborom protokolov je aj „Jabber“.

„Jabber“ sa všeobecne pokladá za „Linux v oblasti IM“. Medzi jeho hlavné prednosti patria:

- *Otvorený formát:* Na rozdiel od komerčných protokolov aké ma napr. ICQ, AIM či MSN, sú Jabber protokoly otvorené, teda ich formát je voľne prístupný k verejnosti.
- *Decentralizovaný:* Prakticky hocikto si môže spustiť svoj vlastný Jabber server a prípadne si ho upraviť podľa svojich potrieb.
- *Extensibilný:* Vďaka použití XML sa môžu Jabber protokoly rôzne upravovať. Hlavne toto je vlastnosť, ktorú vo svojej práci využívam.

[3] Pre zariadenia s konfiguráciou CLDC. Zariadenia s CDC majú CVM – Compact Virtual Machine.
[4] „K“ ako „Kilo“
[5] IM - „instant message“

1.3 Motivácia a cieľ práce

Táto práca je predovšetkým zameraná na spojenie mobilných telefónov do piconet, na nahrávanie a posielanie médií pomocou „Bluetooth“. Ide o to, preskúmať možnosti takéhoto spojenia a vyriešiť problémy ako je napr. smerovanie dát, čo „Bluetooth“ v čase robenia tejto práce nie je schopný samostatne riešiť, ako aj zistiť ako sa pokročilo s možnosťami JSR-135. Zároveň sa pridala možnosť posielať dáta cez Jabber, aby sa ukázala extensibilita jeho protokolov a zvýšila celková užitočnosť aplikácie. Média, ktoré možno posielať, sú text, zvuk, fotky a video. Samozrejme, posielať možno len také média, ktoré mobilný telefón podporuje.

Treba zdôrazniť, že cieľom práce nebolo vytvorenie aplikácie, ktorá by reálne fungovala na všetkých mobilných telefónoch. To by si vyžadovalo veľké investície do komerčných emulátorov ako aj zabezpečenie mobilných telefónov, čo všetko presahuje moje časové aj finančné možnosti.

Cieľom bolo naprogramovať aplikáciu, ktorá sa v prípade potreby dá bez väčších problémov upraviť a urobiť funkčným pre rôzne mobilné telefóny. Toto som chcel dosiahnuť tak, že som prácu skúšal na emulátore. Detaily vytváranie aplikácie a prácu s emulátorom som rozpracoval v kapitole 4.

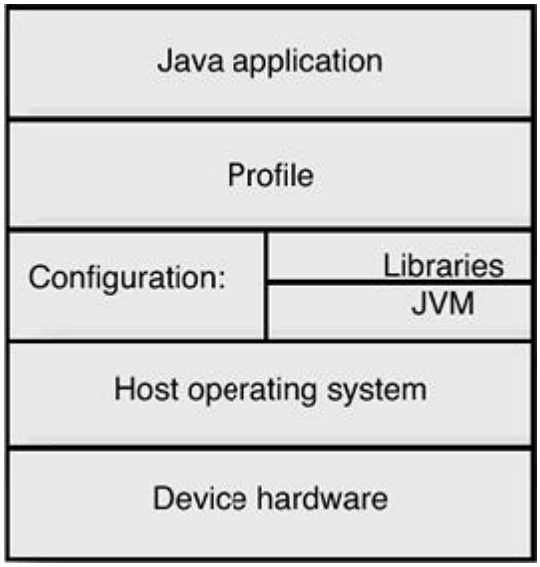
1.4 Obsah práce

Po úvode sa zvoľna ponorím do konfigurácii a profilov platformy J2ME v kapitole 2. Nasledovať bude kapitola 3, kde v krátkosti zhrniem technológiu Bluetooth. V kapitole 4 sa venujem postupu a úskaliam vyvíjania tejto aplikácie. Kapitola 5 je venovaná záverečnému zhodnoteniu práce. V prílohe „A“ sa nachádza užívateľská dokumentácia a v prílohe „B“ je programátorská dokumentácia. Prajem príjemné čítanie.

Kapitola 2

J2ME konfigurácie a profily

Vzhľadom na rôznorodosť zariadení, pre ktoré bola platforma J2ME vytvorená, sa vývojári rozhodli urobiť J2ME modulárnu pomocou konfigurácii a profilov. Pre objasnenie tu máme obrázok.



Obr. 1 – J2ME Stack [1]

1.1 Úvod do konfigurácii a profilov

Konfigurácia súvisí hlavne s hardvérom. Zariadenia v jednej konfigurácii majú podobné pamäťové a výkonnostné vlastnosti. Taktiež špecifikuje minimum vymožeností[1], ktoré musia zariadenia mať. Skladá sa z troch častí:

- *Sada podporovaných Java knižníc a API*
- *Vlastnosti JVM (KVM) (CVM)*
- *Vlastnosti jazyka J2ME*

Aby strata kompatibility nebola až taká veľká, tak boli vyvinuté zatiaľ len dve konfigurácie, a to CLDC a CDC.

[1] z angličtiny “features”

CDC: Connected Device Configuration. Je určená pre zariadenia s pamäťou presahujúcou[2] 2 MB. Používa Compact Virtual Machine (CVM). CDC je konfiguráciou skôr pre PDA a zariadenia s väčšou výkonnosťou. Nie je použitá v mobilných telefónoch, aj keď v tomto čase už sú na trhu PDA s možnosťou telefonovania. Tým naznačujem, že sa postupne stiera hranica, pri ktorej sa prestáva implementovať CDC a je vhodnejšie použiť CLDC. Konfigurácii CLDC bude venovaný odsek 1.2.

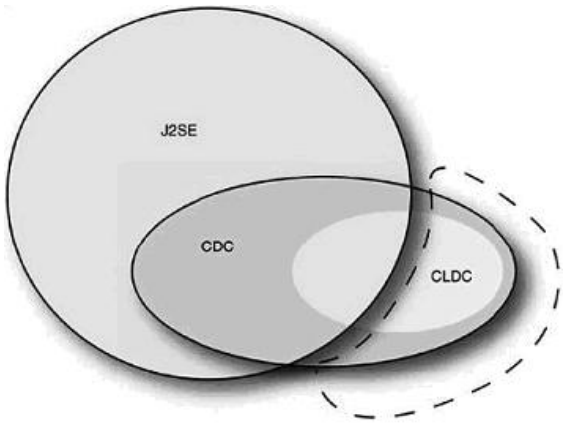
Profil je, ako vidno z Obr. 1, postavený nad konfiguráciou. Profily obsahujú knižnice, ktoré bližšie špecifikujú charakteristiku zariadení, ktoré ich majú implementované. FP[3] je profil, pomocou ktorého je možné vytvoriť si vlastný profil. Najviac používaný profil je Mobile Information Device Profile.

1.2 CLDC

Charakteristika CLDC zariadení je následovná:

- *160KB až 512KB pamäte pre J2ME*
- *16 bitový alebo 32 bitový procesor*
- *nízke nároky na energiu, batériou napájané zariadenia*
- *obmedzená vysielacia kapacita*

Pre lepšiu predstavu o tom, ako knižnice CLDC zapadajú do Java balíčkov[4] pripájam ďalší obrázok.



Obr. 2 – CLDC knižnica [1]

Z Obr.2 je jasne vidieť, že okrem niektorých pôvodných Java knižníc ma aj také, ktoré boli doprogramované práve kvôli odlišnej architektúre mobilných zariadení.

[2] RAM aj ROM
[3] FP – Foundation profile
[4] Tie balíčky a objekty v java.* namespace možno či nemožno použiť. (Balíček – package)

Tab. 1 – balíčky CLDC 1.0	
CLDC Balíček	**Popis**
java.io	IO balíčky z platformy J2SE
java.lang	Triedy a rozhrania[5] pre prácu s VM. Takisto pochádza z platformy J2SE
java.util	Rôzne nástroj z platformy J2SE
javax.microedition.io	CLDC Connection framework

V čase písania tejto práce sú k dispozícii dve verzie CLDC špecifikácii, a to verzie 1.0 a 1.1. Rozdiel medzi nimi je inkrementálny, teda v 1.1 sa nachádzajú niektoré triedy a metódy, ktoré v 1.0 nie sú. V aplikácii som sa rozhodol použiť verziu 1.1 práve preto, že zefektívňujú programovanie.

1.3 MIDP

MIDP[6] je profil postavený nad CLDC. Vzhľadom na to, že CLDC je veľmi všeobecná konfigurácia, v ktorej je obsiahnutých veľa rôznych zariadení, je potrebné vytvoriť profily, ktoré by umožnili na nich vyvíjať aplikácie. MIDP patrí medzi najznámejšie a najpoužívanejšie profily. Prvá verzia, ktorá vznikla, je MIP 1.0

Charakteristika zariadení, pre ktoré bol MIDP 1.0 vyvinutý je zhruba nasledovná:

- *veľkosť obrazovky minimálne 96x65 pixelov*
- *minimálne 1-bitové farby*
- *klávesnica alebo dotykový displej*
- *128 KB pamäte pre MIDP komponenty*
- *8 KB pre perzistentné[7] dáta*
- *32 KB pamäte pre Java heap*
- *obojsmerný bezdrôtový prenos dát*

Keďže vymoženosti MID zariadení je naozaj veľa, MIDP 1.0 sa zameriava pre možnosť vývoja len niektorých z nich a to:

- *ovládanie a výzor aplikácii*
- *užívateľské rozhranie*
- *ukladanie perzistentných dát*
- *správa siete*
- *práca s časom*

[5] z angličtiny „interface“
[6] MIDP – Mobile Information Device Profile
[7] dáta, ktoré ostávajú v pamäti aj po ukončení aplikácie

Tab. 2 – balíčky MIDP 1.0	
MIDP Balíček	**Popis**
javax.microedition.lcdui	Triedy a rozhrania[8] pre UI[9]
javax.microedition.rms	Podpora pre perzistentné ukladanie dát
javax.microedition.io	MIDP Connection framework
javax.microedition.midlet	MIDP podporné typy
java.io	IO balíčky z platformy J2SE
java.lang	Triedy a rozhrania pre prácu s VM. Takisto pochádza z platformy J2SE
java.util	Rôzne nástroj z platformy J2SE

Z Tab. 2 je vidieť, že MIDP 1.0 obsahuje aj balíčky CLDC. Pre svoju prácu som však potreboval využiť MIDP 2.0, ktorý okrem funkcii MIDP 1.0 implementuje mimo iného aj podporu pre nahrávanie a prehrávanie médií (JSR-135). Samotné JSR-135 však musí byť podporované mobilným telefónom, aby ho bolo možné využiť.

[8] z angličtiny “interface”
[9] UI – “user interface”

Kapitola 3

Bluetooth

Ako som už v úvode naznačil, „Bluetooth“ je bezdrôtový komunikačný protokol pracujúci vo frekvenčnom rozsahu 2,4 GHz. Bol vyvinutý ako klient-server architektúra, teda funguje tak, že sa vytvorí „Bluetooth“ server a k nemu sa pripájajú „Bluetooth“ klienti.

V roku 1999 vznikla prvá verzia špecifikácie Bluetooth, a to verzia 1.0. Kvôli rôznym nejednoznačnostiam však boli niektoré vlastnosti implementované v každom zariadení inak. To viedlo k tomu, že sa v decembri roku 2000 tieto nejednoznačnosti spísali a upravili do ďalšej špecifikácie 1.0b. Navyše sa umožnilo vypínať niektoré funkcie, aby sa šetril zdroj energie. Po ďalších úpravách vznikla vo februári 2001 špecifikácia 1.1. V čase písania tejto práce je najnovšia verzia 2.1, ktorá zdvojnásobuje prenosovú rýchlosť[1]. Túto aplikáciu vyvíjam pre Bluetooth verziu 1.1, no keďže je zvykom Bluetooth SIGu robiť verzie spätne kompatibilné, netreba sa v najbližšom čase obávať nefunkčnosti.

V tejto kapitole bližšie rozoberiem jeho špecifiká a jeho vzťah k J2ME.

1.1 Bluetooth a 802.11b

802.11b je, podobne ako Bluetooth, bezdrôtový komunikačný protokol pracujúci na rovnakej frekvencii[2]. Bol vyvinutý skôr než Bluetooth a má oproti nemu niekoľko výhod, ako napr. vyššia prenosová rýchlosť[3], schopnosť prenášať veľké dátové súbory či väčší dosah[4].

Avšak 802.11b nemôže byť použitá pre komunikáciu s periférnymi zariadeniami ako myš, klávesnica či tlačiareň. Takisto je nevhodná na prenos malých dátových súborov či hlasový prenos. Navyše spotrebuje veľké množstvo energie, zatiaľ čo Bluetooth má spotrebu až o päťstokrát menšiu.

[1] na 2,1 Mb/s
[2] podľa jednej štúdie (http://www.forrester.com) sa zariadenia pracujúce s Bluetooth a 802.11b navzájom nerušia.
[3] 11 Mb/s
[4] cca 100 metrov

1.2 Bluetooth zariadenia

Ako som načrtol v predošlom odseku, Bluetooth pracuje v rádiovom frekvenčnom rozsahu 2,400 GHz – 2,483 GHz. To je rovnaký rozsah aký používa mikrovlná rúra. Z toho dôvodu je časť verejnosti znepokojená nad možnými zdravotnými následkami. Podľa výkonu sa Bluetooth zariadenia delia na tri triedy:

Tab. 3 – Triedy Bluetooth zariadení		
Trieda	Vyžarovací výkon	**Dosah**
Trieda I	*100 mW*	100 metrov
Trieda II	*2,5 mW*	20 metrov
Trieda III	*1 mW*	10 metrov

Bluetooth v mobilných telefónoch[5] či iných zariadení napájané batériou patrí zvyčajne do triedy III, čo je výkonovo menšie než vysielanie mobilného telefónu, teda netreba mať kvôli Bluetooth zvýšené obavy o zdravie.

Zariadenia sa ďalej delia na PP[6] a PMP[7]. PP zariadenia vedia komunikovať len s jedným Bluetooth zariadením. PMP vedia komunikovať s viac zariadeniami naraz[8].

Obr. 3 - Point to Point [2]

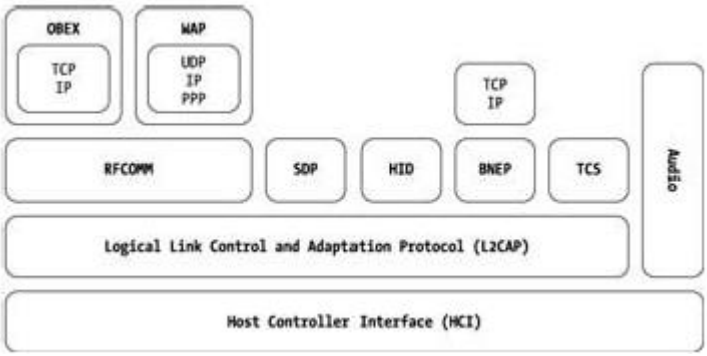
PtP zariadenia bývajú zväčša slúchadlá s mikrofónom, klávesnice, tlačiarne alebo myši, teda periférne zariadenia.

[5] Netreba si mýliť Bluetooth vysielanie s vysielaním mobilného telefónu. Až Bluetooth triedy I je výkonovo približne rovnaké vysielaniu mobilného telefónu.
[6] PP – Point to Point
[7] PMP – Point to Multi-point
[8] zvyčajne naraz so 7 inými zariadeniami

Obr. 4 - Point to Multipoint [2]

1.3 Bluetooth protocol stack

„Bluetooth protocol stack" je sada protokolov, ktoré slúžia na to, aby bolo možné ovládať Bluetooth zariadenia a aby bolo možné komunikovať s inými zariadeniami. Teda umožňuje programovanie aplikácii, ktoré využívajú Bluetooth hardvér. Je rozdelený na vrstvy, pre lepšie pochopenie prikladám obrázok.



Obr. 4 - Bluetooth protocol stack [2]

Vzhľadom na to, že tých protokolov/vrstiev nie je najmenej, popíšem len tie, ktoré vo svojej práci využívam.

Host Controller Interface (HCI)

Toto rozhranie má na starosti dorozumievanie sa Bluetooth hardvéru s hardvérom prístroja, na ktorom je nasadený, napríklad keď je Bluetooth vysielač pripojený na USB port počítača, alebo je pevne pripájkovaný na hardvér mobilného telefónu.

Logical Link Control and Adaptation Protocol (L2CAP)

Cez L2CAP vrstvu pretekajú všetky dáta, ktoré sa cez Bluetooth posielajú. Výnimkou je len zvuk, ktorý má priamy prístup k HCI, aby nevznikalo pri telefonovaní veľké zdržanie. Využíva sa pri známom „Bluetooth headset“. Tu sa veľké kusy dát segmentujú pri odosielaní, a naopak spájajú pri príjmaní. Takisto je L2CAP schopné príjmať naraz dáta z viacerých protokolov, ktoré sú nad ním.

Server Discovery Protocol (SDP)

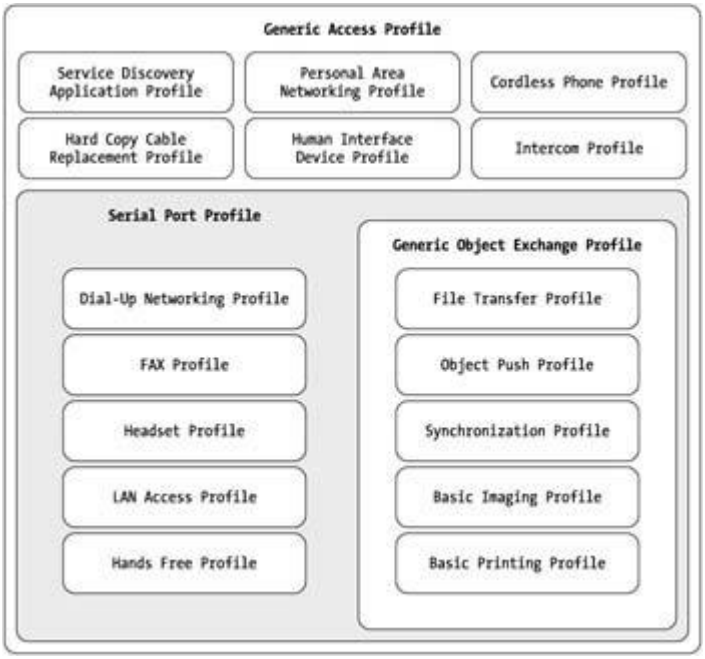
SDP slúži, ako anglický názov napovedá, na objavovanie služieb. Funguje to tak, že sa Bluetooth klient „porozhliadne“ po okolitých Bluetooth zariadení - serverov, ktoré majú zapnutú službu s nejakým názvom, ako napr. telefón môže poskytovať službu „Posielanie SMS“.

Radio Frequency Communication (RFCOMM)

RFCOMM ma nahradzovať sériové porty, resp. simuluje funkčnosť sériového portu. Dá sa povedať, že slúži ako náhrada za kábel ku sériovému portu.

1.4 Bluetooth profily

Ako som naznačil v predošlom odseku, „Bluetooth protocol stack“ slúži na to, aby sa vôbec dal Bluetooth využívať, aby bol funkčný. Avšak na funkčnosť sa nevzťahujú záruka kompatibility s inými Bluetooth zariadeniami. Na zaručenie kompatibility slúžia profily. Koncept nie je veľmi odlišný od J2ME profilov, funguje na rovnakom princípu, no je dôležité si ich medzi sebou nemýliť. Opäť uvediem obrázok a potom krátky popis profilov, ktoré používam.



Obr. 5 - Závislosti profilov [2]

Generic Access Profile (GAP)

Základný profil, používaný na objavovanie a vytváranie spojenia s inými Bluetooth zariadeniami. Obsahuje aj základné funkcie pre zabezpečenie komunikácie. Ako je možné vidieť z obrázku, každý profil nutne využíva funkcie GAP.

Service Discovery Application Profile (SDAP)

Tento profil priamo pracuje s SDP vrstvou v „Bluetooth protocol stack“. Slúži na hľadanie služieb.

Serial Port Profile (SPP)

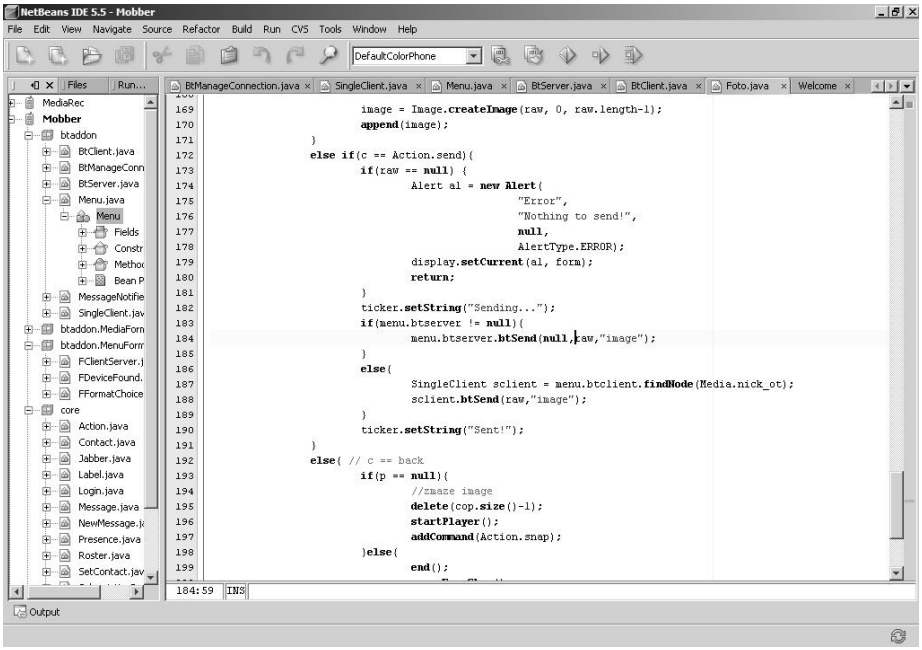
SPP využíva RFCOMM. Vytvára virtuálny port na Bluetooth zariadení, cez ktorý komunikuje práve pomocou RFCOMM. Operačný systém pokladá takýto virtuálny port za obyčajný sériový port.

Kapitola 4

Vytváranie MIDletu a súvisiace problémy

Na vytváranie MIDletu sú potrebné dva programy: textový editor, v lepšom prípade vývojové prostredie na písanie kódu a emulátor mobilného telefónu.

Kvôli jednoduchšiemu debuggingu som si vybral „Netbeans“ IDE[1] s „Mobility Pack“. Netbeans je freevér vývojové prostredie naprogramované v jazyku Java. Samotné Netbeans však na vývoj MIDletov nestačí. Treba si k tomu nainštalovať „Mobility Pack“, čo je prídavný modul slúžiaci práve tomuto účelu.



Obr. 6 - Netbeans IDE s Mobility Pack

Netbeans s Mobility Pack síce umožňuje vývoj samotného MIDletu, no nemá vstavaný emulátor mobilného telefónu. Ten treba inštalovať osobitne.

Čo sa týka emulátorov, má človek na výber veľa variant, komerčné aj freevér. Vybral som si emulátor, ktorý sa nachádza v Sun Java Wireless Toolkit. Je to sada nástrojov, pomocou ktorých sa dajú vyvíjať aplikácie pre zariadenia, ktoré spĺňajú JSR – 185, teda bezdrôtové zariadenia. Obsahuje nástroje pre vytvorenie MIDletu, pomocné nástroje ako napríklad sledovanie využitia pamäti[2], emulátor a hlavne podporuje všetky JSR, ktoré používam.

[1] IDE – vývojové prostredie
[2] tzv. „memory monitor“

1.1 Proces vytvorenia MIDletu

Vytvorenie MIDletu pozostáva z niekoľkých fáz, ktoré teraz popíšem.

Programovanie

Je jasné, že pred hocijakou manipuláciou aplikácie ju treba naprogramovať v nejakom programovacom jazyku. Z toho vzniknú súbory so zdrojovým kódom. Treba mať urobenú určitú adresárovú štruktúru. V hlavnom adresári treba mať nasledovné podadresáre:

- */bin :* tu sa nachádza *.jad, *.jar a MANIFEST.MF súbor
- */classes :* Skompilované a predverifikované *.class súbory
- */res :* obrázky, ktoré MIDlet používa a iné
- */src :* zdrojové kódy MIDletu
- */tmpclasses :* skompilované *.class, ktoré ešte neprešli verifikáciou

Takáto štruktúra je štandardná, no nie je záväzná. Niektoré vývojové prostredia môžu používať aj inú štruktúru.

Kompilácia

Kompiláciou zo súborov so zdrojovým kódom vzniknú *.class súbory, teda Java bytecode, v adresári */tmpclasses*, keďže ešte nie sú verifikované.

Predverifikácia

Predverifikuje *.class súbory v /tmpclasses a upravené ich umiestni do adresára /classes. Dôvodom je ten, že CLDC verifikácia v mobilnom telefóne využíva vďaka inému algoritmu oveľa menej pamäte než typická J2SE verifikácia. K tomu však potrebuje informácie, ktoré sa v procese predverifikácie vkladajú do *.class súborov ešte predtým, než sa MIDlet nainštaluje do mobilného telefónu.

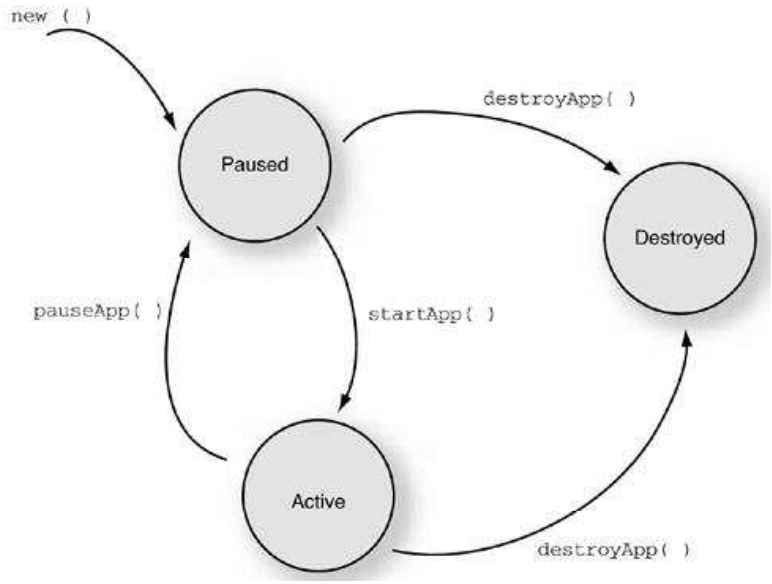
Balenie

Tento proces spočíva v zabalení MIDlet súborov do Java archívu, teda *.jar súboru. Ďalej sa môže vytvoriť manifest súbor, *.MF, ktorý popisuje obsah *.jar súboru. Iný súbor, *.jad, popisuje samotný MIDlet. Síce sa v niektorých knihách píše, že je „voliteľný[3]“, veľa emulátorov a aj mobilných telefónov si ho pri inštalovaní MIDletu vyžaduje.

[3] z angličtiny „optional“

1.2 Základná štruktúra MIDletu

Každý MIDlet má rovnaký životný cyklus, ktorý dobre ilustruje nasledovný obrázok.



Obr. 7 - Životný cyklus MIDletu [1]

Teda môže sa nachádzať len v troch stavoch[4]. Celý cyklus riadi AMS.

Poznámka o AMS

„AMS“ je skratka pre „Application Management System“. Je to softvér, z užívateľského hľadiska „neviditeľný“, ktorý sa stará o inštaláciu, spustenie, ukončenie a odinštalovanie MIDletu. AMS sa dodáva už inštalovaný v mobilnom telefóne. Vyvíja v jazyku C či Java.

1.3 Nahrávanie pomocou MMAPI

V krátkosti zhrniem nahrávanie médií v J2ME pomocou Mobile Media API (JSR – 135).

Na ovládanie médií sa používa inštancia triedy *Player*, ktorú získame zavolaním funkcie **createPlayer(string data_source)**. Premenná „data_source“ určuje ako sa bude *Player* používať.

Napríklad príkaz `Player p = Manager.createPlayer(“capture://audio”);` vytvorí *Player*, ktorý umožní nahrávanie zvuku.

[4] môže sa nachádzať ešte v jednom stavu a to keď ešte nie je MIDlet spustený

Z triedy *Player* sa môže získať niekoľko druhov ovládaní pomocou funkcie **getControl(String controlType)** ako napríklad VideoControl, MIDIControl, RecordControl či VolumeControl. Tieto ovládania sú veľmi intuitívne pomenované triedy, s ktorými sa dajú ovládať jednotlivé vlastnosti médii.

Predtým, než je možné *Player* vôbec využiť, musí prejsť niekoľkými stavmi. Východzí stav po vytvorení je *UNREALIZED*. Z tohoto stavu sa dostane do stavu *REALIZED*, kedy získava informácie o zdrojoch, ktoré bude potrebovať. V ďalšom stave *PREFETCHED* sa tieto zdroje získajú a v stave *STARTED* sa tieto zdroje používajú. Zo stavu *STARTED* možno prejsť do stavu *REALIZED* či *PREFETCHED* a zo všetkých stavov možno prejsť do stavu *CLOSED*, kedy *Player* odovzdá zdroje a už sa nedá opätovne použit. Príslušné funkcie majú podobné názvy: **realize()**, **prefetch()**, **start()**, a **close()**.

Zavolaním na *Player* po start po jeho vytvorení implicitne zavolá aj funkcie, ktoré stavu *STARTED* prechádzajú, čo môže spôsobiť oneskorenie v nahrávaní, či prehrávaní. Preto je väčšinou dobré uviesť Player do stavu *REALIZED* aby bol čas medzi tým čo uživateľ napríklad zapne nahrávanie a samotné spustenie nahrávania čo najkratší.

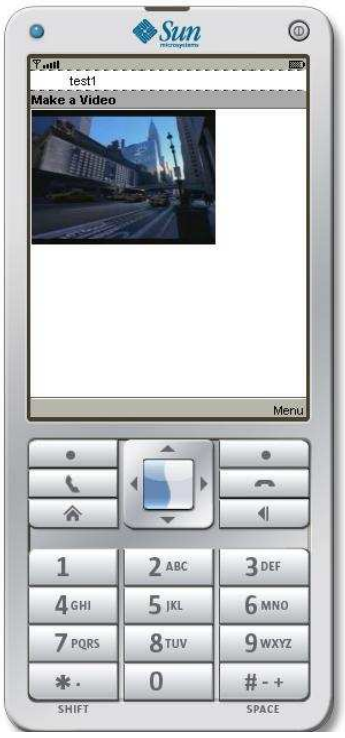
1.4 Emulátor verzus Mobilný telefón

V priebehu práce mi začalo byť jasné, že s emulátorom, ktorý používam, nie je možné vytvoriť MIDlet, ktorý by bol funkčný na všetkých mobilných telefónoch. Toto zistenie ma celkom prekvapilo, lebo som sa domnieval, že keď emulátor aj mobilné telefóny podporujú rovnaké JSR, tak že nebude nutné MIDlet opakovane nahrávať a skúšať aj na mobilnom telefónu. Uvediem pár príkladov.

- Emulátor pri odosielaní dát cez Bluetooth vyprázdňuje buffer odchádzajúcich dát implicitne, nie je nutné používať funkciu **flush()**, no v mobilnom telefóne sa inak dáta nepošlú. Teda po nainštalovaní MIDletu na požičaný mobilný telefón posielanie textu nefungovalo. Táto zdanlivo drobná chyba spôsobila trojdňové zdržanie. Nakoniec som to vyriešil tak, že sa mi podarilo nájsť zdrojový kód ukážkového MIDletu, ktorý bol zameraný čisto na posielanie textu cez Bluetooth. Posielanie spravené podobne, navyše sa používala iba spomínaná funkcia **flush()**. Vzhľadom na to, že som mal k dispozícii iba jeden mobilný telefón, nemôžem s istotou povedať, či sa takto správajú aj iné telefóny.

- Väčšie problémy sa týkajú JSR-135, teda Mobile Media API. Napriek širokým možnostiam JSR-135 a možnosť nainštalovať kameru na počítač nie je možné v čase písania tejto práce použiť emulátor na zachytávanie obrazu ako takého. V Sun emulátore v2.2 sa zobrazila šablóna, na novom Sun emulátore v2.5.1 sa prehráva ukážkové video, ktoré sa samo spustí. Uvediem obrázok oboch verzii emulátorov.

Sun WTK emulátor v2.2



Sun WTK emulátor v2.5.1

Zavádzajúci je fakt, že na emulátore sa video nedá nahrávať napriek tomu, že použitím funkcie **Manager.getSupportedContentTypes(string protocol)** sa dosadením do premennej „protocol“ dá „capture“, teda vypísanie všetkých MIME[5], ktoré možno zachytiť či nahrať pomocou daného zariadenia.

- Pri vytváraní fotiek treba použiť iné MIME vo funkcii **createPlayer(string data_source)**. Zatiaľ čo emulátor musí mať v premennej „data_source“ hodnotu „capture://video“, čo vlastne nie je nahrávanie videa, ako som spomenul v predošlom bode. Mobilný telefón potrebuje mat hodnotu „capture://image“[6]. Opäť nie je zaručené, že iné telefóny sa budú správať rovnako.

- Nedostatok pamäti je veľmi nepríjemným obmedzením u mobilných telefónoch. V mojej aplikácii sa to prejavovalo pri UUENCODE, keď som alokoval bajtové pole s dátami, nechtiac som alokoval dve takéto polia, čo bolo navyše 70KB a stačilo na zahlásenie OutOfMemoryError. Aj keď sa mi podarilo nájsť chybu a odstrániť ju, otázkou ako riešiť nedostatok pamäti. V emulátore sa dá nastaviť veľkosť heapu či perzistentnej pamäti, no u mobilných telefónoch sú tieto údaje pevne dané. V programe som to urobil tak, že sa dá nahrávať maximálne 10 sekúnd a nikdy nie je viac ako jedno alokované bajtové pole určené na nahrávanie médií v pamäti.

[5] MIME – „Multipurpose Internet Mail Extensions“. Je to vlastne popis médii. Pre zoznam registrovaných MIME médii viď. http://www.iana.org/assignments/media-types/
[6] konkrétne telefón značky Nokia 6280

- Pri hľadaní služieb pre Bluetooth klienta alebo ponúkaní služieb pre Bluetooth server sa používa LIAC[7] a GIAC[7]. Sú to časy, ktoré určujú približne ako dlho má klient hľadať okolité Bluetooth zariadenia s nejakou službou, respektíve ako dlho má Bluetooth server ponúkať nejakú službu. Aj na mobilnom telefóne značky Nokia 6230 aj na emulátore vznikli problémy buď s tým, že sa služba hľadá príliš krátku dobu (LIAC má trvať približne 60 sekúnd) alebo že služba po očakávanom skončení bola stále ponúkaná.

- Chyby v implementácia JSR-82 v Sun emulátore. Dôsledok je napríklad taký, že ak sa klient spustí pred serverom, tak klient už nenájde server.

1.5 Bluetooth meshed network

„Mesh networking“ je spôsob ako smerovať dáta, hlas a inštrukcie medzi zariadeniami, v tomto prípade medzi mobilnými telefónmi. Idea bola taká, že sa vytvorí Bluetooth scatternet a v takejto sieti by sa posielal text, zvuk, fotky a video medzi telefónmi. Bluetooth scatternet je sieť vytvorená z navzájom prepojených Bluetooth klientov a serverov.

Toto však na **mobilných telefónoch** s J2ME v Bluetooth špecifikácii 1.1 nie je možné urobiť, a to z dvoch nasledujúcich dôvodov.

- *Bluetooth klient sa nemôže pripojiť naraz na dva Bluetooth servery*
- *Nemôže byť naraz zapnutý Bluetooth server aj Bluetooth klient*

Tieto obmedzenia sa nedajú obist. Príčina je tá, že naprogramovaný Bluetooth server alebo klient má exkluzívny prístup k Bluetooth hardvéru. Teda nie je možné, aby napríklad v jednom vlákne MIDletu bežal Bluetooth server a v druhom Bluetooth klient, lebo vždy by malo len jedno vlákno prístup k hardvéru a druhé nie. Teoreticky by bolo možné, aby telefón striedal svoju funkciu ako klient a server na určité časové obdobie. Takého úvahy nie sú až tak nereálne no problém s MIDletmi je ten, že za každým, čo sa spustí Bluetooth server, alebo sa MIDlet ako Bluetooth klient chce pripojiť k Bluetooth serveru, tak žiada užívateľa o súhlas. Tento mechanizmus sa mi napriek snahám nepodarilo obísť. Avšak verím, že je len otázkou času, kedy sa tieto obmedzenia v ďalších špecifikáciách odstránia.

Aby som mohol napriek tomu implementovať špecifikáciu bakalárskej práce, obmedzil som vytvorenie siete na Bluetooth piconet, čo je sieť pozostávajúca z jedného servera a maximálne sedem klientov, ktorí sú na tento server pripojení. Komunikácia medzi serverom a klientom prebieha priamo, komunikáciu medzi dvoma klientmi som musel vyriešiť sám, keďže zatiaľ neexistuje Bluetooth protokol alebo profil, ktorý by takúto komunikáciu umožnil. Ako bónus som pridal automatické opätovné zoskupenie telefónov do siete v prípade, že vypadne server. Podrobný popis môjho riešenia nájdete v prílohe „B“ – programátorská dokumentácia.

[7] LIAC – Limited Inquiry Access Code, GIAC – General Inquiry Access Code

Kapitola 5

Záver

1.1 Budúcnosť Mobberu

Mobber bol vyvinutý ako Jabber komunikátor pre mobilné telefóny. Pridaním svojho Bluetooth modulu som viac než zdvojnásobil jeho veľkosť a teda ho spravil reálne nepoužiteľným pre staršie mobilné telefóny, ktoré vyžadujú, aby bola veľkosť MIDletu maximálne 128KB. Pri práci som si uvedomil, že efektívnejšie riešenie by bolo rozdeliť túto prácu na dva samostatné MIDlety. Pôvodný Mobber by si ponechal iba nahrávanie médií a uuencode/uudecode, teda by ostala možnosť posielať média cez Jabber (samozrejme, iba medzi dvoma telefónmi, ktoré majú kompatibilnú verziu Mobbera) a vyvíjal by sa osobitne od Bluetooth modulu.

Bluetooth modul by sa v prípade rozšírenia technológie o spomínanú možnosť súbežného spustenia servera a klienta ďalej mohol rozvíjať a použiť pre plnohodnotný mesh network ako aj za krátky čas upraviť pre použitie na ľubovolnom mobilnom telefóne podporujúci príslušné špecifikácie.

1.2 Záverečné zhodnotenie

Bluetooth a J2ME sú neustále vyvíjajúce sa technológie, s ktorými je náročné pracovať. Okrem algoritmických problémov sa programátor často stretáva s nekompatibilitou, ktorá sa ťažšie rieši.

Vytvoril som MIDlet, ktorý reálne na emulátoru vie vytvoriť Bluetooth piconet a medzi jednotlivými inštanciami emulátorov posielať text, zvuk a obraz. Zvuk aj obraz možno posielať cez protokol Jabber pomocou uuencode/uudecode. Zdrojový kód pre nahrávanie a posielanie videa som napísal a síce je podobný s nahrávaním zvuku, nemám možnosť ho odskúšať, keďže emulátor nepodporuje nahrávanie videa a k najnovším mobilným telefónom ako napríklad séria s60 od firmy Nokia.

Vlastnosti výsledného softvéru tejto práci sa líšia od pôvodnej špecifikácie, no stálo za tú námahu ho vyvinúť. Slúži totižto aj ako výstraha všetkým programátorom, ktorí sa rozhodnú vyvíjať aplikácie pre mobilné telefóny a hlavne tam kde sa pracuje s rýchlo sa rozvíjajúcimi sa technológiami. Pri vývoji sa má sústrediť na úzky profil mobilných telefónov s veľmi podobnými špecifikáciami, najlepšie od jednej firmy a nie na veľa rôznych druhov naraz.

Zdroje

[1] Piroumian V. (2000): *Wireless J2ME Platform Programming*, Prentice Hall PTR.

[2] Hopkins B., Antony R. (2003): *Bluetooth for Java*, Apress.

[3] Goyal V.: J2ME Tutorial
http://today.java.net/pub/a/today/2005/02/09/j2me1.html
http://today.java.net/pub/a/today/2005/05/03/midletUI.html
http://today.java.net/pub/a/today/2005/09/27/j2me4.html

[4] Mahmoud Q. H.: Wireless Application Programming with J2ME and Bluetooth
http://developers.sun.com/techtopics/mobility/midp/articles/bluetooth1/

[5] Mahmoud Q. H.: Part II: The Java APIs for Bluetooth Wireless Technology
http://developers.sun.com/techtopics/mobility/midp/articles/bluetooth2/

[6] Grasza G.: Mobber – mobile jabber communicator
http://mobber.gryf.info/

[7] Sun Microsystems, Inc.: Java APIs for Bluetooth Wireless Technology (JSR-82)
http://jcp.org/en/jsr/detail?id=82

[8] Nokia Corp.: Games over Bluetooth
http://forum.nokia.com/info/sw.nokia.com/id/2b17fb6f-b9a4-4cd8-80fd-94b8251a048e/Games_Over_Bluetooth_v1_0_en.zip.html

[9] Nokia Corp.: Mobile Media API Developer's Guide
http://forum.nokia.com/info/sw.nokia.com/id/f4f9f446-c335-4e23-ab39-5c49b003f202/MIDP_Mobile_Media_API_Developers_Guide_v2_0_en.pdf.html

[10] Jabber
http://www.jabber.org

[11] Java Community Process
http://jcp.org/en/home/index

[12] Uuencode - Wikipedia
http://en.wikipedia.org/wiki/Uuencode

[13] Giguere E. (2002): The Importance of the Mobile Information Device Profile (MIDP)
http://www.ericgiguere.com/articles/importance-of-midp.html

[14] Sun Java Wireless Toolkit for CLDC
http://java.sun.com/products/sjwtoolkit/

Prílohy

Príloha „A“ – Užívateľská dokumentácia

Cieľom dokumentácie je oboznámiť vás s funkciami tohto softvéru a naučiť vás ho používať. Informácie o spôsobe fungovania sa nachádzajú v prílohe „B“ – programátorská dokumentácia.

Čo to dokáže:

- Posielať text, zvuk, fotky a video cez Bluetooth a cez Jabber[1].
- Vytvorenie piconetu z okolitých Bluetooth zariadení a vzájomné posielanie medzi nimi.

Popis aplikácie:

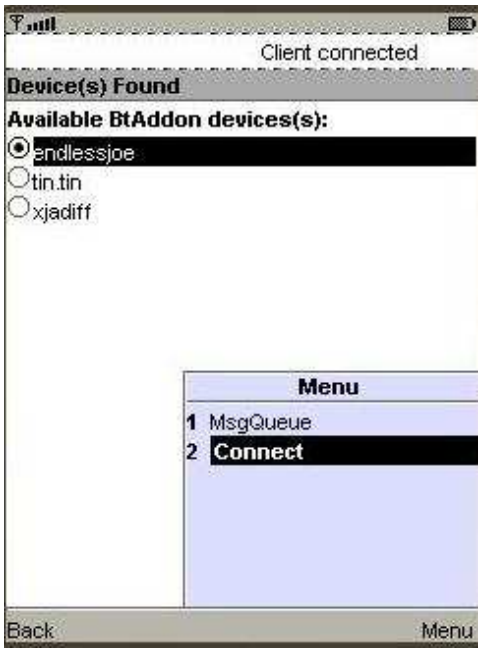
Po spustení sa zobrazí prihlasovací formulár.

Stlačením na Menu si môžete vybrať buď *Login*, čo vás prihlási na vaše Jabber konto alebo *BTAddon*, čo vám zobrazí možnosti Bluetooth modulu.

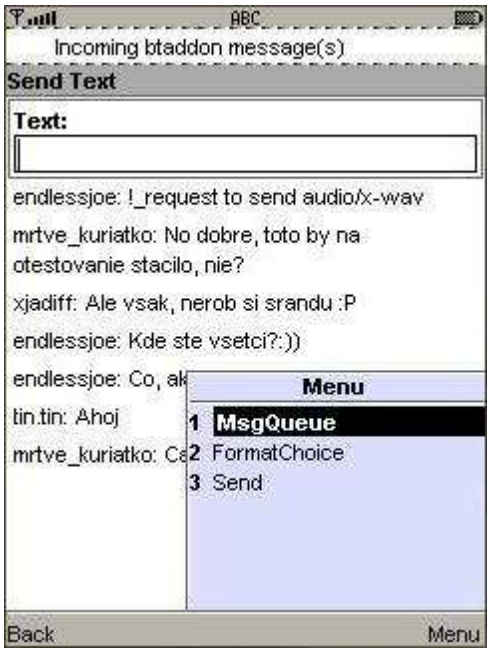
Pozor pri výbere mena. Jabber ID má štandardný formát, no pre *BTAddon* sa berie meno (nickname) len do zavináča (@). Zadávanie hesla nemá na *BTAddon* žiaden vplyv.

[1] Pre tie mobilné telefóny, respektíve emulátory, ktoré to podporujú

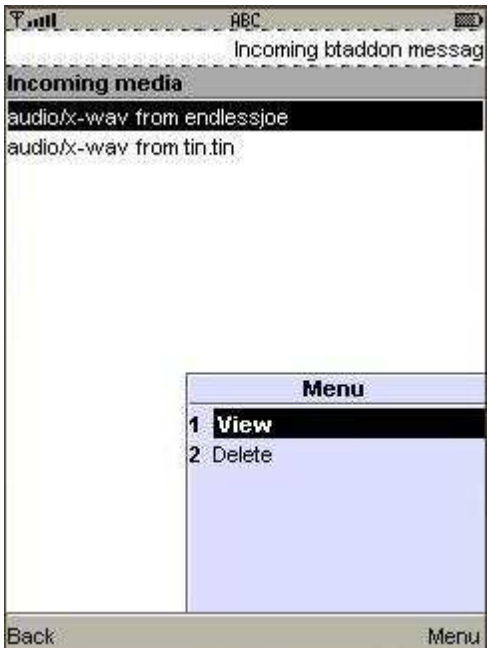
Pozrieme sa teraz bližšie na *BTAddon*. Po odkliknutí je nám daný na výber *Client*, *Server* alebo *Jabber-record*, ktorý teraz nefunguje, lebo nie ste ešte prihlásený na vaše Jabber konto. Ak nemáte spustený už nejaký server, vyberte si Server a kliknite na *OK*. Aplikácia sa vás možno opýta, či môže vytvoriť Bluetooth server spojenie. Odkliknite *Yes*.



Teraz stačí len čakať, než sa nejakí klienti pripoja. Ak chcete nejakému klientovi poslať nejaké média, označte jeho meno a kliknite na *Connect*. Text sa posiela každému klientovi aj serveru, takže text každý vidí.



Takto vyzerá Bluetooth chat. Napíšete správu, kliknite na *Menu*, vyberiete *Send* a vaša správa sa rozpošle. Správy začínajúce znakmi „!_“ značia, že vám niekto chce poslať zvuk, fotku či video. Rozpoznáte to podľa MIME. Ak chcete média prijať alebo naopak, neprijať, kliknite na *MsgQueue*. Ak chcete človeku, ktorého ste si pre stlačením *Connect* označili, poslať média, vyberte si *FormatChoice*, kde sa vám zobrazí zoznam médií, ktoré mu môžete poslať.



V *MsgQueue*, čo je vlastne fronta správ, respektíve požiadaviek, si môžete pozrieť kto vám chce aký mediálny formát poslať. Môžete si vybrať *View* a pozrieť si ho, alebo *Delete*, teda ho neprijmete. V tom prípade bude dotyčnému zaslaná správa, že ste odmietli médium prijať. Výberom *Back* sa vrátite k chatu.

Vo *FormatChoice* máte na výber formáty, ktoré vy môžete nahrávať a zároveň adresát vášho média môže prehrávať. V Sun emulátore v2.5 máte de facto na výber len dva funkčný formát, a to audio/x-wave a jeden mnou doplnený image/emulator.

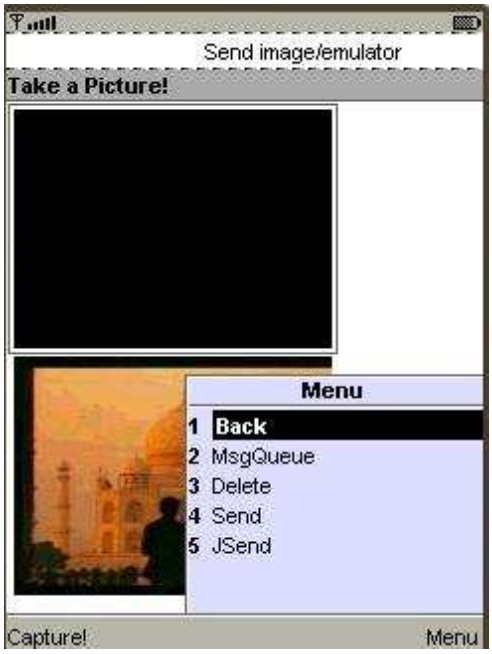
Síce vám to ukáže aj formát video/vnd.sun.rgb565, tak vedzte, že je to chyba emulátora a nemožno nahrávať video. Ak sa aj pokúsite nahrávať, bude vypísaná chyba. Na telefónoch ako napríklad séria 60 od firmy Nokia by malo byť možné video nahrávať.

Zvuk a video majú podobné ovládanie, preto po vysvetlení nahrávania a prehrávania zvuku pochopíte aj nahrávanie a prehrávanie videa. Na výber máte päť možností.

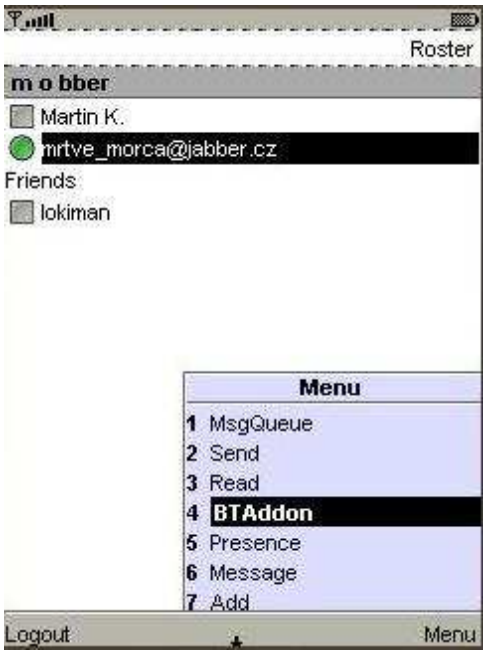
- *Record*: spustí nahrávanie.
- *Play*: spustí prehrávanie nahraného zvuku/videa, prípadne zahlási, že nič nie je nahrané.
- *MsgQueue*: Zobrazí spomínanú frontu správ.

Send: pošle požiadavku o prijatí nahrávky. Ak sa vo výberoch vrátite a vyberiete si ako adresáta vašich nahrávok niekoho iného, pôvodný adresát nedostane vašu nahrávku.

- *JSend:* slúži na posielanie cez Jabber, ak ste prihlásený na svoje konto a telefón, na ktorý to chcete poslať, tiež má spustený tento MIDlet. Bližšie informácie v ďalšom obrázku.



Robenie fotiek je veľmi jednoduché. Stačí stlačiť „Capture!“ a fotka sa vám zobrazí pod kamerovým displejom. Fotky nie je možné ukladať. „Delete“ zmaže aktuálnu fotku. V Sun emulátoru sa nespustí kamera, ale sa začne v nekonečnom cykle prehrávať ukážkový videoklip, z ktorého môžete zachytávať obraz, teda akoby ste na fyzickom telefóne s kamerou robili normálne fotky. Ostatné funkcie sa zhodujú s tými, čo som vysvetlil pri popisovaní nahrávania a prehrávania zvuku a videa.



V prípade, že sa na začiatku prihlásite na svoje Jabber konto, budete mať na výber okrem funkcii typických pre Jabber IM klienta navyše aj dve možnosti.

Môžete si pozerať *MsgQueue* tak, ako som to vyššie popisoval. Ďalej tu máte *BTAddon*, čo som už tiež popisoval, teda máte rovnaké možnosti. Navyše môžete použiť *Jabber-record* na posielanie zvuku, fotiek či videa cez Jabber. Opäť pripomínam, že telefón, na ktorý to posielate musí mať spustený tento MIDlet. Môžete si aj posielať textové správy, ktoré v prípade, že ste pripojený aj ako Bluetooth klient či server, budú tiež všetci vidieť.

Pozor si treba dávať vtedy, keď náhodou budete prihlásený na Jabber konto a budete sa chcieť prihlásiť aj na Bluetooth sieť, pričom prvá časť vášho JID bude zhodná s menom nejakého Bluetooth klienta či servera. Potom vám bude pochopiteľne zobrazená chyba, že meno, s ktorým sa chcete prihlásiť, je už obsadené.

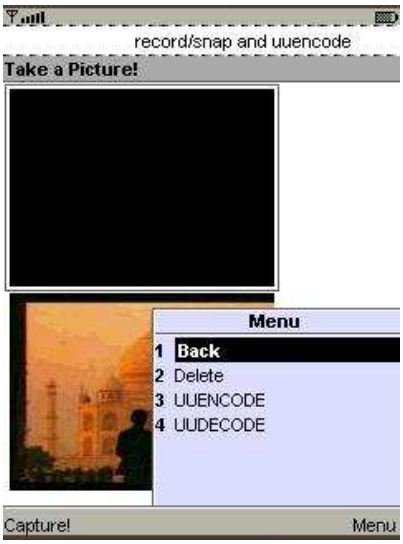
Keďže v MIDletu nastala (nie mojou vinou) situácia, kedy nie je možné na emulátore[2] odskúšať posielanie médií cez Jabber pomocou uuencode, vytvoril som pomocný MIDlet, ktorý nahráva a prehráva zvuk a robí fotky a vzniknuté média konvertuje pomocou uuencode z *byte array* do *String* a cez uudecode naspäť do *byte array*, ktorý možno prehrať ako zvuk či zobraziť ako fotku.

Tento MIDlet som vytvoril preto, aby som dokázal, že uuencode/uudecode, ktorý som naprogramoval, skutočne funguje.

[2] Sun emulátor v2.5 sa nachádza na pribalenom CD. Pre informácie o obsahu CD viď. prílohu „C“ – obsah CD.

Popis pomocnej aplikácie:

Formáty zvuku a fotiek sa vyberajú rovnako ako v Mobberu. Zmena nastala v menu, kde som pridal možnosti „UUENCODE“, „UUDECODE“ a „Delete“.



Po vybraní možnosti „UUENCODE“ sa zobrazí informácia o reťazci, ktorý reprezentuje zakódované dáta. „Delete“ slúži na to, aby sa nahrané dáta zmazali. „UUDECODE“ reťazec rozkóduje naspäť na pole bajtov, ktoré možno prehrávať.

Príloha „B“ – Programátorská dokumentácia

V tejto dokumentácii bude popísané ako tento MIDlet funguje. Pre informácie o používaní tohto MIDletu, prečítajte si prílohu „A“ – užívateľská dokumentácia.

Zmeny oproti Mobberu:

Vzhľadom na to, že som upravoval existujúci MIDlet, nasleduje zoznam zmien oproti pôvodnému MIDletu.

- Balíčky *btaddon, btaddon.MediaForm* a *btaddon.MenuForm* a ich obsah som pridal sám.
- Do balíčka *utils* som pridal súbory *Person.java* a *Uuencode.java*
- Robil som úpravy na súboroch *Login.java, Roster.java* a *Jabber.java*

Popis behu aplikácie:

Každý MIDlet má typický životný cyklus, ktorý som krátko popísal v odseku 1.2 kapitole 4 a predpokladám, že čitateľ je oboznámený s prvkami GUI používané v J2ME. Začnem s pridaným Bluetooth modulom.

Hlavnou triedou v tomto modulu je *Menu*. Tu majú inštancie všetky triedy dediace triedu *Form*, teda tie, ktoré sa zobrazujú užívateľovi. *Menu* sa stará aj o výmenu zobraziteľných tried. Je tu aj ovládanie odpájania Bluetooth serveru či klienta a jeho zapínanie. Ako prvá trieda sa zobrazí trieda *FClientServer* s výberom Bluetooth servera či klienta. Po nej nasleduje buď *FDeviceFound* alebo keď si vyberiete Jabber-record, tak *FFormatChoice*, čo je trieda na výber formátov.

So zobrazením *FDeviceFound* sa zároveň vytvorí a spustí nová inštancia triedy *BtServer* alebo triedy *BtClient*, ktoré sú podedené od triedy *Thread*. Tieto triedy opíšem neskôr.

FDeviceFound slúži buď pre Bluetooth server na zistenie telefónov, ktoré sa pripojili alebo pre Bluetooth klienta na vypísanie zoznamu serverov a zároveň, keď sa pripojí na nejaký server, na výpis telefónov (mien) v tom piconete.

Po vybratí ktorému telefónu sa budú posielať média sa prepne do triedy *Text*. Trieda *Text* sa stará o zobrazovanie správ a o posielanie správ podľa toho, či je užívateľ pripojený ako server alebo ako klient. Pozrie sa či je pripojený na Jabber konto a ak áno, tak pošle text aj cez Jabber, ale nie ako normálnu Jabber správu, ale ako BTAddon správu, o ktorej sa tiež zmienim neskôr. V triede *Text* sa dá dostať do triedy *MessageNotifier* alebo *FFormatChoice*.

Do triedy *MessageNotifier*, ktorá dedí po triede *List*, prichádzajú požiadavky na prijatie médií od telefónov. Tieto požiadavky ukladám ako vnorenú triedu *Msg* do vektoru, ktorá obsahuje meno žiadateľa, mediálny formát a index, teda referencia potrebná na prípadné odstránenie správy z vektoru. Metódy na odstránenie, pridanie a hľadanie daného požiadavku sa tu taktiež nachádzajú ako aj možnosť prijať požiadavku, čo zobrazí v prípade, že sa médium ešte dá poslať, príslušnú triedu, teda buď *AudioVideo* alebo *Foto*.

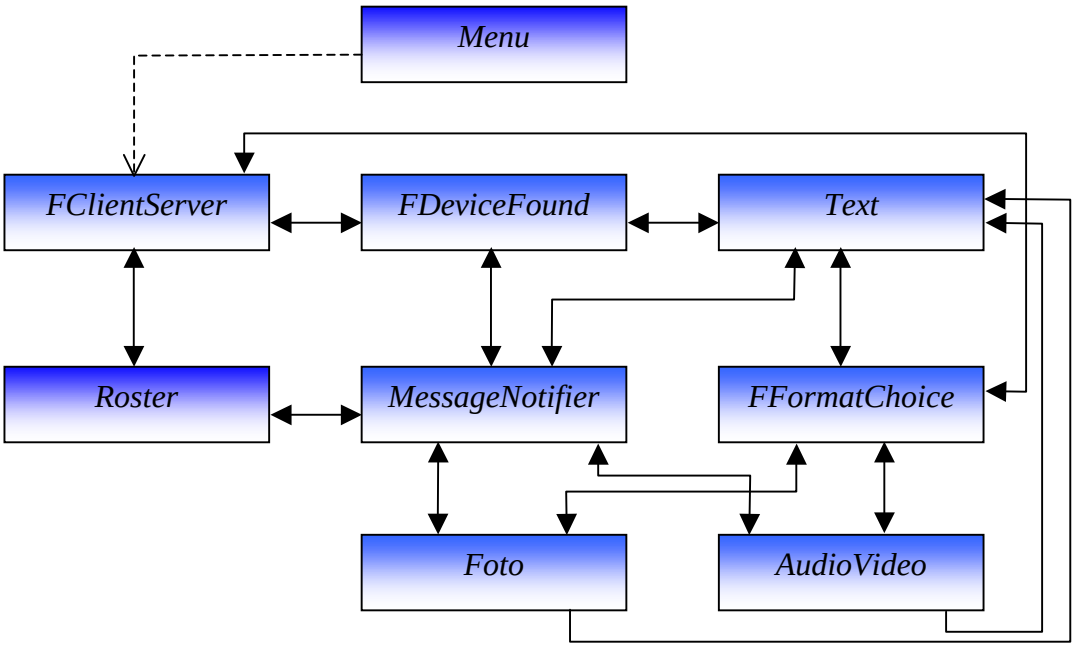
Vrátim sa k spomínanému *FFormatChoice*. V tejto triede si môžete prezerať média, ktoré je možné adresátovi poslať. Zoznam sa tu nahrá počas pripájania sa na piconet, čo vysvetlím v ďalšom odseku. Nech je formát médií akýkoľvek, môžete sa dostať do jednej z dvoch tried, a to *AudioVideo* alebo *Foto*. Na výber sú na Sun emulátoru tri formáty, z čoho formát „image/emulator“ je mnou doplnený. Tomuto formátu som dal predponu „image“ preto aby bolo jasné, že sa jedná o zachytávania obrazu a príponu „emulátor“, aby program vedel, že keď bude inicializovať *Player*, aby namiesto lokátoru „capture://image“, čo funguje pre mobilné telefóny, dal ako lokátor „capture://video“, čo inicializuje Player pre Sun emulátor.

AudioVideo je trieda, kde sa dá nahrávať zvuk alebo video. Na nahrávanie používam vnorenú triedu *StopRec* podedenú po *Thread*, na prehrávanie vnorenú triedu *PlayRec*, tiež podedenú po *Thread*. Vzhľadom na to, že média sú veľmi citlivé na chyby, aj tá najmenšia závada môže spôsobiť Exception, preto sa tu nachádza veľa try-catch blokov[1]. Na uskladnenie nahrávky slúži pole bajtov byteArray. *Foto* je trieda štruktúrovo podobná triede *AudioVideo*, teda keď sa zorientujete v *AudioVideo*, *Foto* tiež pochopíte.

[1] Typický catch blok v tejto práci pozostáva z vytvorenia inštancie triedy *Alert*, do ktorej sa vloží chybové hlásenie, jej zobrazenie na displej pomocou metódy **FormShow()** triedy *Menu* a často aj následné prepnutie na takú zobraziteľnú triedu, z ktorej sa dá pokračovať v používaní aplikácie.

Schéma zobrazovania foriem aplikácie:

Pre objasnenie dostupnosti zobrazovacích tried prikladám následovnú schému.



Menu som pridal preto, že ako prvé zobrazí *FClientServer* z BTAddon. *Roster* je trieda, ktorá sa zobrazí keď sa prihlásite na svoje Jabber konto a z nej sa podľa schémy pristupuje k triedam Bluetooth modulu, preto je tam čiarkovaná šípka. Schému nemožno pokladať za graf dostupnosti foriem, šípky sú navzájom nezávislé.

Aby bolo možné komunikovať cez Jabber s Bluetooth modulom, som pridal do existujúcej XML schémy Jabber správy vlastný prvok <btaddon>, ktorý má niekoľko atribútov.

type a jeho hodnoty:

- ”negotiate”
- ”decided”
- ”enough_mem”
- “request”
- “request_accepted”
- “request_time_out”

len_cap: udáva počet formátov, ktoré možno mobilným telefónom zachytávať.

len_all: udáva počet formátov, ktoré možno na mobilnom telefóne prehrávať.

Ďalej som ešte pridal prvok <format> s jedným atribútom:

type a jeho hodnoty:

- “capture”
- “all”

Celé to funguje tak, že keď je užívateľ v *Roster* a v menu klikne na BTAddon, tak sa pošle Jabber správa vybranému človeku v kontakt liste, v ktorom je zoznam formátov, ktoré môže nahrávať a prehrávať. Keďže správa obsahuje prvok <btaddon>, tak sa správa nezobrazí ako textová správa, ale zachytí a začne spracovávať program. V prípade, že adresát takejto správy nemá túto verziu upraveného Mobberu, tak sa mu zobrazí iba prázdna správa. Ak však túto verziu má, načíta si formáty, do svojho *FFormatChoice* vloží len tie formáty, ktoré môže nahrávať a odošle naspäť formáty, ktoré môže prehrávať. Urobí sa prienik množiny formátov, ktoré dotyčný môže prehrávať s množinou formátov, ktoré sa môžu nahrávať a vložia sa do *FFormatChoice*. Ako prostredník tu slúži trieda *Person*, ktorá sa stará o uuencode správ a ich posielanie, ako aj nastavovanie formátov.

Posielanie médií sa posiela pomocou uuencode/uudecode (textu sa posiela štandardne). Uuencode je algoritmus, ako pole bajtov konvertovať na reťazec. Uudecode robí presný opak.

UUENCODE/UUDECODE:

Algoritmus prebieha tak, že pole bajtov sa doplní o nuly tak, aby bola jeho dĺžka deliteľná číslom 3. Potom sa pole spracováva po troch bajtoch tak, že sa tri bajty rozdelia do štyroch skupín po šiestich bitoch, pričom každá šestica sa vyjadrí decimálnou hodnotou, ku ktorej sa pripočíta číslo 32. Toto číslo sa uloží ako znak do reťazca. Takto po spracovaní vznikne reťazec znakov, ktorých najmenšia hodnota je 32 (medzera) a najväčšia je 96 (apostrof). Tento reťazec sa potom vloží do správy a pošle štandardným spôsobom cez Jabber, kde sa na druhom konci reťazec načíta a prevedie sa naspäť do poľa bajtov. Toto sa robí tak, že sa vezme štvorica znakov z reťazca a pomocou bit-shift operácii sa prevedú do výsledného bajtového poľa.

Original characters	C				a				t			
Original ASCII, decimal	67				97				116			
ASCII, binary	0 1 0 0 0 0				1 1 0 1 1 0				0 0 0 1 0 1			
New decimal values	16				54				5			52
+32	48				86				37			84
Uuencoded characters	0				V				%			T

Obr. 8 - uuencode [12]

Ako posledná časť prílohy „B“ nasleduje popis Bluetooth komunikácie servera a klienta.

Bluetooth komunikácia servera:

Po vybraní si servera z výberu triedy *FClientServer* sa spusti nové vlákno starajúce sa o serverovú časť komunikácie. V tomto vlákne sa nastaví Bluetooth hardvér aby bol telefón viditeľný okolitým Bluetooth zariadeniam a spustí sa Bluetooth služba identifikovaná jedinečným UUID. Služba sa nastaví ako voľná a začne čakať na pripojenie klienta vykonaním metódy **acceptAndOpen()**, ktorá je blokujúca, preto sa server spúšťa v osobitnom vlákne.

Ak sa nejaký klient pripojí, nadviaže sa s ním spojenie a toto spojenie, teda DataOutputStream a DataInputStream, spolu s informáciami o klientovi sa uloží do inštancie triedy *SingleClient* a táto trieda sa vloží do vektoru s_Clients a začne sa čakať znova. Položka s indexom 0 v s_Clients je vždy server, teda aj server má v tejto položke informácie o sebe samom, akurát sa nespúšťa vlákno v tejto inštancií.

Trieda *SingleClient* je podedená po triede *Thread* a akonáhle sú informácie o klientovi v nej načítane, tak spusti metódu **ManageConnection()** inštancie triedy *BtManageConnection*. Táto metóda sa stará o príjmanie a interpretovanie správ, ktoré klienti serveru pošlú.

Po nadviazaní spojenia s klientom nastáva výmena možností nahrávania a prehrávania formátov podobne ako som vyššie popísal pri komunikácii v Jabberu. Navyše sa tu ešte posiela novému pripojenému klientovi zoznam mien ostatných klientov pripojených k serveru spolu s ich formátmi. V prípade, keď sa zistí odpojenie klienta, tak server pošle informáciu, že sa dotyčný klient odpojil, zvyšným klientom.

Bluetooth komunikácia klienta:

Trieda *BtClient* tiež je podedená od triedy *Thread* a spúšťa sa tak, ako sa spúšťa server. Najprv začne vyhľadávať okolité zapnuté Bluetooth zariadenia. Kvôli zrýchleniu hľadá najprv také, čo má uložené v pamäti, potom tie, na ktoré bolo často pristupované (tiež ich má v pamäti) a nakoniec sa začne rozhliadať po okolí. Zariadenia, ktoré nájde, ukladá do vektoru „discoveredDevices“. Po skončení vyhľadávania a za predpokladu, že sa našlo aspoň jedno zariadenie, začne sa na nájdených zariadeniach hľadanie služieb. Hľadám službu s určitým UUID, ktoré mám k dispozícii. Je to rovnaké UUID, ktorým som službu na Bluetooth serveru spustil. Keď sa hľadanie služieb na zariadeniach skončí, vypíšu sa len tie zariadenia (telefóny), ktoré majú spustenú túto službu. Keď sa vyberie nejaký server, prebehne už spomínaná výmena formátov a vypíše sa zoznam mien (klientov a servera piconetu).

Na rozdiel od servera si klient neukladá informácie do vektoru s_Clients spojení s ostatnými klientmi, keďže nemá ako im priamo posielať dáta. Jedine so serverom sa udržuje spojenie. Z tohto dôvodu sa posielanie dát medzi klientmi rieši preposielaním cez server.

Posielanie textu má okamžitý priebeh. Klient pošle správu serveru, ktorý ju prepošle všetkým klientom. Klient, ktorý správu odoslal, si sám filtruje správy, aby si tú istú správu dvakrát nezapísal. Posielanie médií medzi klientom a serverom či klientom a iným klientom je podobné ako som popísal posielanie médií cez Jabber. Keď klient posiela médium serveru, pošle najprv požiadavku, server-užívateľ sa rozhodne či ju chce prijať. Ak áno, pošle potvrdenie o tom, že chce médium prijať, na čo klient pošle médium. Predtým sa však skontroluje, či je na serveru ako cieľovom zariadení dostatok pamäte. Ak je, všetko prebehne bez problémov. Ak nie, prenos dát sa nekoná a chybová hlásenie o nedostatku pamäti na cieľovom telefóne sa zobrazí klientovi. Serveru sa v tickeru tiež zobrazí správa o nedostatku pamäte.

Ak klient chce poslať inému klientovi médium, tak komunikácia je rovnaká až na to, že sa komunikuje cez server a to tak, že keď klient zistí, že v danej inštancií triedy *SingleClient* nie je informácia o spojení, tak sa posielajú dáta spolu s menom klienta serveru. Serveru sa pošle navyše reťazec „route“, ktorý keď zachytí, tak vie, že sa jedná o preposielanie a prepošle dáta cieľovému klientovi.

Pomocná aplikácia:

Pomocná aplikácia, ako je spomenuté v užívateľskej dokumentácii, slúži ako dôkaz, že moja implementácia uuencode/uudecode funguje a keďže som už vysvetlil princíp algoritmu ako aj beh triedy *AudioVideo*, nie je potrebné nič viac dodať.

Príloha „C“ – Obsah CD

Pribalené CD obsahuje tieto adresáre a súbory:

- */utilities* – tu sa nachádzajú inštalačné súbory vývojového prostredia Netbeans v5.5.1, ďalej Netbeans Mobility Pack, ktorý umožňuje vyvíjať v Netbeans IDE aplikácie v J2ME a Sun Java Wireless Toolkit v2.5, ktorý obsahuje aj emulátor mobilného telefónu.

- */src* – tu sa nachádzajú zdrojové kódy výslednej práce.

- */bin* – tu je výsledný MobberBTAddon.jar spolu s MobberBTAddon.jad, pripravený na inštaláciu do mobilného telefónu.

- */mobber* – pre porovnanie prikladám pôvodný Mobber. Tento adresár obsahuje zdrojové kódy, súbory mobber.jar a mobber.jad.

- */UuencodingMedia* – zdrojové kódy, UuencodingMedia.jar a UuencodingMedia.jad súbory pomocnej aplikácie

- *bakalarska_praca.pdf* – Text mojej bakalárskej práce vo formáte PDF.

- *uzivatelska_dokumentacia.pdf, programatorska_dokumentacia.pdf* – Texty užívateľskej a programátorskej dokumentácie, ktoré sa nachádzajú aj v bakalárskej práci ako prílohy.