UNIVERSITATEA TEHNICĂ CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE

PROIECT DE DIPLOMĂ

Absolvent
Berari Iulia Dana

Anul 2008

UNIVERSITATEA TEHNICĂ CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
SECŢIA CALCULATOARE

VIZAT DECAN
Prof. Dr. Ing. Sergiu NEDEVSCHI

VIZAT ŞEF DE CATEDRĂ
Prof. Dr. Ing. Rodica Potolea

Medii mobile colaborative.Un algoritm de rutare no-cost pentru retele ad-hoc

Absolvent: **BERARI IULIA DANA**

1. CONŢINUTUL PROIECTULUI:

Introducere, Studiu bibliografic, Concepte introductive, Tehnologii pentru medii mobile, Framework-ul Peer2Me, Implementarea algoritmului de rutare, Dezvoltarea unui MIDlet, Aplicaţia PeerMessenger, Testarea aplicaţiei, Evaluarea framework-ului Peer2Me, Concluzii, Direcţii de dezvoltare

2. LOCUL DOCUMENTAŢIEI:

Universitatea Tehnică Cluj-Napoca, Facultatea de Automatică şi Calculatoare, Secţia Calculatoare

3. DATA EMITERII TEMEI: IANUARIE 2008

4. TERMEN DE PREDARE: IUNIE 2008

CONDUCĂTOR DE PROIECT
Şef lucrări. Ing. Cosmina IVAN

SEMNĂTURA ABSOLVENT

Cuprins

Listă Figuri

Partea I

Prezentarea temei lucrării

Capitolul 1. Introducere

Telefonul mobil este echipamentul de comunicare cel mai utilizat în prezent, având un impact tot mai puternic asupra societăţii datorită funcţionalităţilor multiple oferite, funcţionalităţi precum comunicarea la distanţă, realizarea de poze sau utilizarea unei agende elecronice. O dată cu creşterea numărului de telefoane mobile au crescut şi performanţele acestora şi totodată funcţiile de care aceste dispozitive dispun.

Evoluţia telefoanelor mobile este foarte rapidă, modelele noi lansate cu un an în urmă fiind considerate depăşite în prezent datorită faptului că noi functionalităţi şi noi standarde sunt introduse. Potrivit unui studiu dat publicităţii de compania de cercetare Pew Internet Project [1], cele mai populare activităţi, pe lângă cele de recepţionare şi efectuare a apelurilor, sunt cele de transmitere a mesajelor SMS şi fotografierea, 58% dintre repondenţi efectuand ambele activităţi, cel puţin o dată pe zi. Studiul mai arată faptul ca telefonul mobil este echipamentul de comunicare la care oamenii ar renunţa cel mai greu. Telefoanele mobile din ziua de astazi sunt capabile să realizeze sarcini care erau considerate imposibile în urmă cu doi ani, sarcini precum controlul unor procese industriale sau comanda, controlul şi informarea asupra unui sistem de alarmă, motiv pentru care studiul pentru identificarea unor noi funcţionalităţi, precum trimiterea unui mesaj de genul mesajelor sms într-un mod gratuit, reprezintă o mare provocare.

Crearea de soluţii software unice pentru acest domeniu este însă destul de dificilă datorită varietăţii telefoanelor mobile prezente pe piaţă. Au fost identificate o serie de standarde şi soluţii pentru dezvoltarea de software pentru telefoanele mobile (J2ME, Peer2Me, JXTA). De asemenea, telefoanele mobile sunt în general echipate cu diverse tehnologii pentru realizarea schimbului de date în reţea, precum GSM, Bluetooth, IrDa, WLAN, tehnologii care permit conectarea telefoanelor mobile în diferite moduri, contexte şi distanţe, fiecare având avantajele şi dezavantajele sale. Una dintre cele mai utilizate tehnologii de la ora actuală este tehnologia Bluetooth, aceasta asigurând comunicarea între dipozitive mobile aflate în inele de proximitate de aproximativ zece metri.

Ca urmare a studiului bibliografic efectuat, pentru documentarea solicitată de acest proiect, a fost selectat un framework open source relativ bine documentat, numit Peer2Me. Conform definiţiei framework-ului ca fiind „Un set de clase care înglobează un design abstract de soluţii pentru probleme din aceeaşi familie.” [2], framework-ul Peer2Me oferă dezvoltatorilor posibilitatea de a programa aplicaţii pentru telefoanele mobile fară a ţine cont de diferenţele dintre tehnologiile de reţea sau de modul în care sunt trimise pachetele în reţea. Motivul alegerii acestui framework constă în funcţionalitatea pe care acesta o oferă, şi anume existenţa unui mediu pentru dezvoltarea aplicaţiilor mobile colaborative pentru reţele ad-hoc, fapt care a făcut posibilă integrarea unui algoritm de rutare a mesajelor no-cost, numit Scribble-M.

1.1Motivaţia

Telefonia mobilă a stârnit un mare interes în rândul dezvoltatorilor datorită ritmului alert în care evoluează, iar tehnologia Bluetooth reprezintă unul dintre elementele ce caracterizează această evoluţie, constituind un aport important în realizarea schimbului de date dintre dispozitivele mobile , cu predilectie pentru reţele ad-hoc. Caracteristicile cele mai importante ale acestei tehnologii sunt constituite de lipsa conexiunilor cablate dintre dispozitive şi totodată de faptul că este o tehnologie gratuită. De asemenea, prezenţa acestei tehnologii pe un număr mare de dispozitive mobile reprezintă un element important în alegerea tehnologiei pentru realizarea acestei lucrări de diplomă.

Varietatea telefoanelor mobile prezente pe piaţă a făcut ca realizarea de aplicaţii colaborative pentru reţelele ad-hoc să reprezinte un obiectiv greu de realizat. Framework-ul Peer2Me are ca scop facilitarea procesului de realizare a unor astfel de aplicaţii. Utilizănd tehnologia Bluetooth, framework-ul ofera o serie de functionalităţi, precum trimiterea de mesaje sau fişiere, specifice domeniului temei. Fiind un framework open source, extrem de bine documentat, a fost posibilă realizarea unui studiul detaliat cu scopul integrarii unei noi functionalităţii de optimizare şi anume trimiterea unui mesaj gen SMS la destinatari din afara proximităţii dispozitivului expeditor.

1.2 Descrierea obiectivelor

Scopul principal al acestui proiect este acela de a optimiza framework-ul Peer2Me utilizat în dezvoltarea de aplicaţii colaborative destinate telefoanelor mobile , structurate în reţele ad-hoc de tipul Personal Area Network (PAN). Aceasta optimizare constă în implementarea unui algoritm de rutare no-cost numit Scribble-M, algoritm ce reprezintă o

variantă modificată a unui algoritm similar (Scribble), unul dintre cei mai buni sub aspectul performantelor, din clasa algoritmilor manycast, utilizaţi pentru rutare de mesaje în reţelele ad-hoc folosind tehnologia Bluetooth.

Capitolul 2. Studiu bibliografic

În acest capitol vor fi descrise şi analizate proiecte similare framework-ului Peer2Me, proiecte identificate pe parcursul activităţii de cercetare a acestui domeniu şi care din diferite motive au fost cosiderate nepotrivite pentru realizarea temei proiectului.

2.1 JXTA (Juxtapose)

Este un protocol peer-to-peer open source început în anul 2001 de firma Sun Microsoft. Protocoalele JXTA sunt definite ca un set de mesaje XML care permit oricărui dispozitiv conectat la o reţea să schimbe mesaje şi să colaboreze în mod independent de topologia reţelei. JXTA este cel mai dezvoltat framework P2P de la ora actuală şi a fost proiectat pentru a permite unui număr mare de dispozitive (PC, mainframe, PDA, telefoane mobile) să comunice între ele într-o manieră descentralizată.

Datorită faptului că JXTA este bazat pe un set de protocoale XML, poate fi implementat în orice limbaj de la ora actuală (Java Platform, J2ME). Nodurile JXTA crează o reţea virtuală care permite unui nod să interacţioneze cu alte noduri chiar şi atunci când acestea se află în spatele unui firewall, sau utilizează protocolale diferite de transport. În plus, fiecare resursă este identificată cu un ID unic, de 160 de biţi, astfel încât un nod îşi poate schimba adresa cât timp îşi păstrează un număr unic de identificare.

Există două proiecte în curs de dezvoltare pentru utilizarea framework-ului JXTA pe telefoane mobile: JXME cu proxi şi JXME fără proxi. JXME cu proxi se bazează pe un dispozitiv central pe post de proxi între nodurile din reţea dar care nu asigură necesităţiile necesare unei reţele mobile P2P mature. JXME fără proxi este o încercare de a creea o versiune completă a framework-ului JXTA pentru telefoane mobile. Nici o implementare a variantei de JXME fără proxi nu este disponibilă pentru reţele PAN.

2.2 PeerBox

PeerBox este un serviciu peer-to-peer pentru partajarea de fişiere. Cu PeerBox utilizatorul poate partaja poze sau fişiere video cu alţi utilizatori direct de pe telefonul mobil. De asemenea, se pot creea profiluri cu poze, se pot trimite mesaje prietenilor sau se pot viziona fotografiile şi fişierele video ale acestora.

PeerBox-ul a fost dezvoltat în diferite versiuni atât pentru Symbian cât şi pentru J2ME. Totuşi framework-ul nu poate fi folosit pentru reţele ad-hoc datorită faptului că schimbul de fişiere se face prin intermediul Internet-ului.

2.3 BEDD

Bedd este un software dezvoltat şi menţinut de Corporaţia Bedd. Spre deosebire de celelalte proiecte similare cu Peer2Me, proiectul Bedd este un software comercial şi nu un proiect de cercetare sau open source. Acest software a fost lansat în Singapore în Mai 2004.

Prin intermediul acestui software utilizatorul poate schimba profiluri cu utilizatorii din proximitatea lui prin Bluetooth, iar cu restul utilizatorilor prin Internet. Unele dintre aplicaţiile incluse în software-ul Bedd sunt BEDDmates folosit pentru schimbul de profile, BEDDtalk folosit pentru conversaţia între telefoanele care dispun de acest software, BEDDbay folosit pentru adăugarea de anunţuri pentru vânzarea, închirierea, cumpărarea sau schimbul de articole.

2.4 JSR-259: Ad Hoc Networking API

JSR-259 Ad Hoc Networking Api este un produs al comunităţii Java (JPC) iniţiat de vânzătorii de telefoane mobile precum Siemens şi Nokia în Noiembrie 2004. Scopul JSR-ului 259 este acela de a crea un API (Application Programming Interface) standard pentru comunicarea între noduri într-o reţea ad hoc care foloseşte J2ME.

API-ul a fost finalizat în Martie 2006, dar nu este disponibil deocamdată pe telefoanele mobile probabil datorită faptului că implementarea unui nou standard pentru dispozitivele mobile comerciale necesită o perioadă îndelungată de timp.

Partea II

Fundamentare teoretică

Capitolul 3. Concepte introductive

În acest capitol sunt prezentate principalele concepte teoretice legate de framework-ul Peer2Me. Datorită faptului că această lucrare de diplomă reprezintă o continuare a lucrărilor de master realizate la Universitatea de Ştiinţă şi Tehnologie din Norvegia, conceptele teoretice prezentate în cadrul acestui capitol vor avea ca punct de plecare conceptele teoretice prezentate în lucrarea de master realizată de Steinar A. Hestnes şi Torbjørn Vatn. [3]

3.1 Reţele wireless

În general, reţelele wireless se referă la reţelele care utilizează semnalele infraroşii sau undele radio pentru partajarea de informaţii sau resurse între dispozitive. Evoluţia rapidă a tehnologiilor wireless a provocat schimbări dramatice în ultimi ani în acest domeniu. Creşterea popularităţii reţelelor wireless a determinat o scădere rapidă a preţului echipamentelor wireless concomitent cu o accentuată îmbunătăţire a performanţelor tehnice ale acestora.

În continuare vor fi descrise principalele tipuri de reţele wireless, catalogate în funcţie de raza lor de acţiune (Figura 3.1).

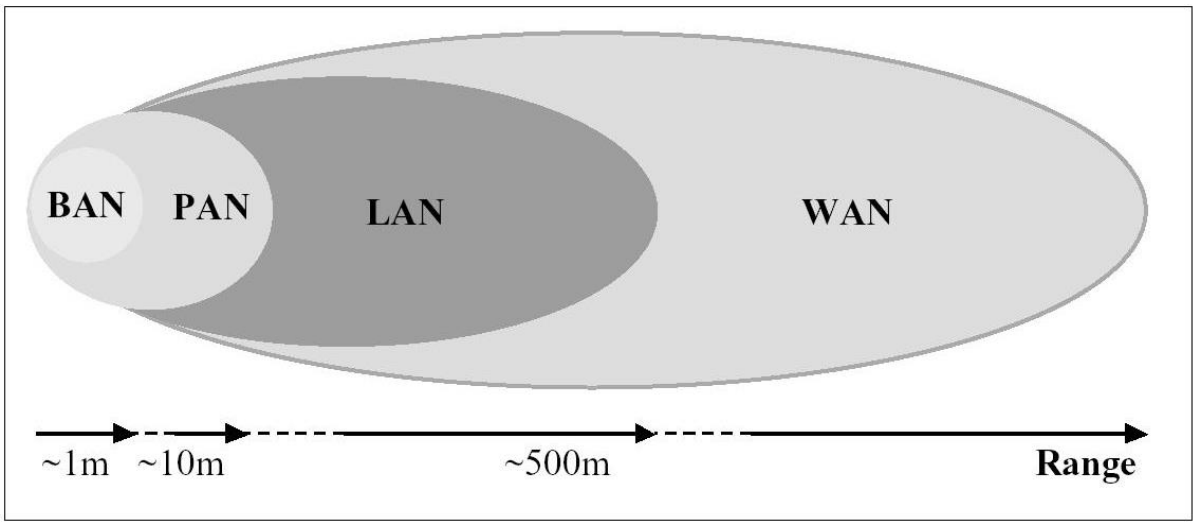

Figura 3.1 Taxonomia reţelelor ad-hoc

Wireless Wide Area Network (WWAN)

Wireless Wide Area Network, denumite în mod uzual WWAN, sunt reţele bazate pe infrastructură, construite de un set de staţii bază care trimit semnale broadcast utilizatorilor de telefoane mobile. Reţelele WWAN adresează necesitatea utilizatorilor de a rămâne conectaţi în permanenţă la reţea, chiar şi atunci când tranversează mari zone geografice, raza de acoperire a reţelei fiind foarte mare. În mod normal conexiunile pot fi realizate peste oraşe, sau chiar peste tări. Astăzi, reţelele folosite pentru astfel de conexiuni sunt chiar reţelele de telefonie mobilă, astfel încât reţele precum GSM (2D) şi UMTS (3D) permit conexiuni wireless aproape în întreaga lume. [4][5]

Wireless Metropolitan Area Network (WMAN)

Wireless Metropolitan Area Network, numite şi WMAN, sunt reţele bazate pe infrastructură, construite de un set de staţii bază. Aceste reţele conectează utilizatori din zone metropolitane, precum o zonă de mai multe clădiri, un campus universitar sau un grup de clădiri pentru birouri. Reţelele WMAN pot fi formate de un număr de transmiţătoare WiFi interconectate, poziţionate astfel încat să asigure zona de acoperire necesitată cu semnale radio. Altfel, pot fi realizate prin utilizarea transmiţătoarelor WiMAX care asigură acoperire wireless pe mai mulţi kilometri, mult mai mare decât WiFi. WiMax are potenţialul de a permite mobilitate wireless pe raza unei zone metropolitane. [5]

Wireless Local Area Network (WLAN)

Wireless Local Area Network, abreviate ca WLAN, sunt reţele care conectează utilizatorii dintr-un spaţiu local, spre exemplu o clădire. Reţelele WLAN pot fi reţele bazate pe infrastructură sau reţele ad-hoc. În cadrul reţelelor WLAN bazate pe infrastructură, staţile wireless se conectează la puncte de acces wireless care definesc o regiune finită de acoperire. Aceste puncte de acces crează o legătură între staţiile wireless şi restul reţelei. Cealaltă alternativă, reţelele ad-hoc, permit utilizatorilor să se conecteze uni cu alţi fără a dispune de o infrastructură fixă cu puncte de acces. Staţiile se conectează unele cu altele în mod direct. Acest tip de reţea se realizeză pentru spaţii foarte limitate, precum o cameră. Exemple de protocoale WLAN tipice includ seria IEEE 802.1, HomeRF şi HiperLAN2. Astăzi, cel mai utilizat protocol de pe piaţa consumatorilor este IEEE 802.11g, care are o rată de date teoretică de 54Mbps. [4][5]

Wireless Personal Area Network (WPAN)

Wireless Personal Area Network (WPAN) conectează utilizatorii de pe raza personală a unui sistem de operare, suportând în mod obişnuit o acoperire de zece metri. Acesta este tipul de reţea pe care o adresează framework-ul Peer2Me. Accentul se pune pe conectarea instantanee dintre dispozitive care administrază date personale sau care facilitează schimbul de date între grupuri mici de indivizi. Un exemplu ar putea fi schimbul spontan de documente şi fisiere de muzică între doi sau mai mulţi indivizi. Un alt exemplu ar putea consta în sincronizarea datelor (de exemplu un calendar) dintre un telefon mobil şi un calculator. Natura acestor şcenarii pentru schimbul de date este ad-hoc şi de cele mai multe ori spontană. Cele mai relevante tehnologii utilizate la ora actuală pentru reţelele WPAN sunt Bluetooth, ZigBee şi infraroşu (IrDA). Datorită faptului că IrDA necesită o conexiune fără obstacole între cele două dispozitive aflate la o distanţă de maxim un metru , Bluetooth-ul este adesea mai adecvat datorită utilizării undelor radio în locul luminii ca mediu de transmisie. Bluetooth-ul utilizeză banda neregulată de 2.4GHz , are o rată de date de maxim 1Mbps şi o rază a semnalului de aproximativ zece metri. Din moment ce framework-ul Peer2Me utilizează tehnologia Bluetooth, mai multe detalii despre această tehnologie şi despre reţelele ad-hoc pot fi găsite în următoarele capitole.

Wireless Body Area Network (WPAN)

O reţea Body Area Network (BAN) este o reţea de componente distribuite pe corpul uman. Acestea pot fi dispozitive diferite precum telefonul mobil, headset, MP3 player etc. fiind conectare printr-o tehnologie wireless.

3.2 Modelul Peer-to-Peer

Modelul Peer-to-Peer presupune creearea unor reţele care se bazează pe puterea de procesare şi laţimea de bandă a tuturor participanţiilor din reţea (peers), în locul utilizării serverelor fixe care să ofere resursele şi servicile necesare reţelei. Idea centrală într-o reţea peer-to-peer este de partajare a resurselor; de a da şi de a obţine ceva de la o comunitate de „semeni”. Nodurile care alcătuiesc o reţea peer-to-peer depind unele de altele pentru obţinerea resurselor şi informaţilor esenţiale sistemului ca ansamblu. Fiecare peer oferă anumite resurse şi obţine altele în schimb. [6]

În figura 3.2 este prezentată o clasificare a sistemelor de calcul în sisteme centralizate şi distribuite. În timp ce sistemele centralizate reprezintă soluţii cu o singură unitate, sistemele

distribuite reprezintă reţele de calculatoare alcătuite din unităţi locale, care comunică şi îşi coordonează acţiuniile prin mesaje. Sistemele distribuite pot fi la rândul lor clasificate în sisteme client-server sau sisteme peer-to-peer. Diferenţa dintre aceste două modele este aceea că modelul client-server dispune de o unitate centrală, un server, care asigură toate serviciile şi resursele necesare reţelei, spre deosebire de modelul peer-to-peer, unde nu există nici o unitate centrală. [6]

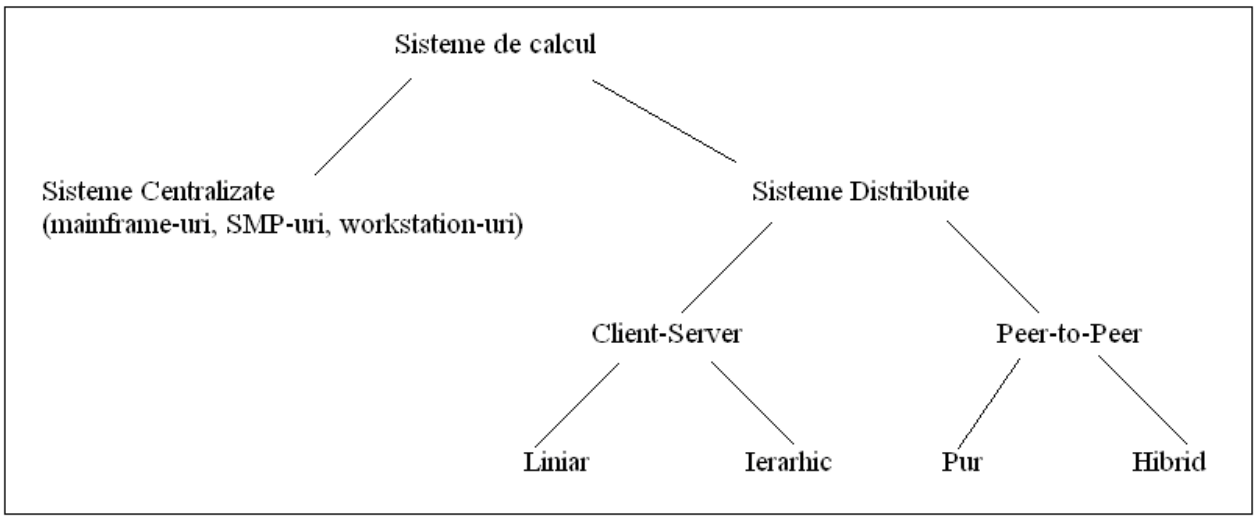

Figura 3.2 Taxonomia Sistemelor de Calculatoare

3.2.1 Clasificarea reţelelor peer-to-peer

Modelul peer-to-peer poate fi pur sau hibrid. Într-un model pur nu există nici o unitate centrală (server) reasponsabilă pentru administrarea sau coordonarea serviciilor şi resurselor din reţea. Toate nodurile reţelei sunt egale şi au aceeaşi responsabilitate în cadrul reţelei. Acest model de reţea peer-to-peer permite nodurilor să intre în reţea sau să părăsească reţeaua în orice moment fără să afecteze comunicarea celorlalte noduri. Un exemplu de reţea pură este Gnutella, un sistem care oferă schimbul de date între calculatoare [6]. Modelul pur de P2P este ilustrat în Figura 3.3.

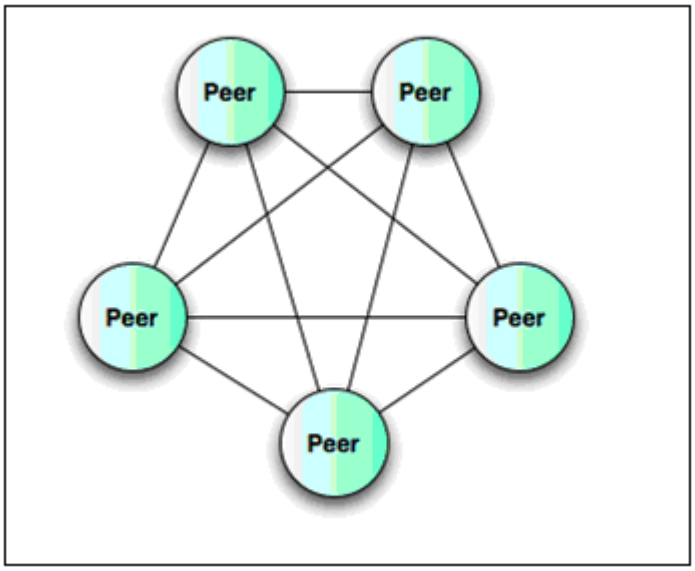

Figura 3.3 Modelul Peer-to-Peer Pur

În modelul peer-to-peer hibrid există de asemenea conexiuni între fiecare nod, dar o unitate centrală asigură anumite servici nodurilor. Figura 3.4 ilustrează un model de P2P hibrid. În acest model, nodurile se conectează mai întâi la server pentru a obţine meta-informaţii, cum ar fi identitatea nodului pe care sunt stocate anumite informaţii sau să verifice accesul la un anumit nod. Exemple de sisteme de calcul care utilizează peer-to-peer hibrid sunt sistemele de partajare a datelor precum Napster şi iMesh. [6]

Alegerea unei arhitecturi peer-to-peer are loc de obicei datorită unor scopuri precum reducerea costurilor, nevoia de îmbunătăţire a performanţelor, îmbunătăţirea scalabilităţii, a dinamismului şi a conumicării ad-hoc. Datorită faptului că toate nodurile unei reţele peer-to-peer alocă resurse, inclusiv lăţime de bandă, spaţiu de memorare şi putere de procesare, odată cu creşterea numărului de noduri, creşte şi capacitatea sistemului. Acestă afirmaţie nu este adevărată în cazul modelului client-server cu un număr fix de servere, în care adăugarea unor noi clienţi ar putea însemna un transfer de date mai scăzut pentru toţi utilizatorii. Natura distribuită a reţelelor peer-to-peer creşte de asemenea robusteţea în cazul eşecurilor, prin replicarea datelor în mai multe noduri şi permiterea nodurilor să găsească datele fără a se baza pe utilizarea unui server centralizat în acest scop. Ca urmare a acestui fapt într-o arhitectură peer-to-peer pură nu va exista niciodată un singur punct de eşec în sistem.

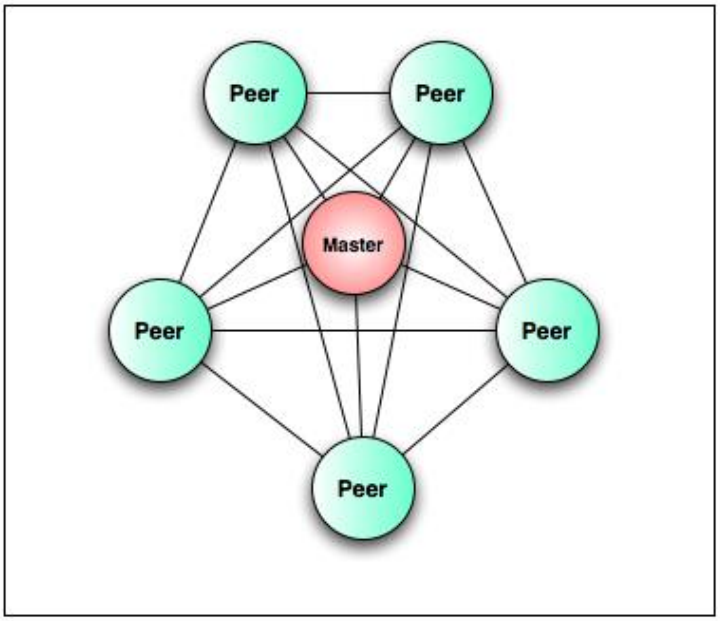

Figura 3.4 Modelul Peer-to-Peer Hibrid

Reţele peer-to-peer sunt utilizate în mod uzual pentru conectarea nodurilor prin conexiuni ad-hoc. Astfel de reţele sunt folosite în realizarea unor procese precum partajarea fişierelor video şi/sau audio în format digital. Modelul peer-to-peer este folosit şi în cadrul aplicaţiilor pentru transmiterea mesajelor şi este foarte adecvat pentru comunicarea în timp real datorită faptului că nu se bazează pe nici un server în primirea şi trimiterea datelor.

3.3 Reţele Mobile Ad-hoc

Reţelele ad-hoc mobile, cunoscute sub numele de MANET, sunt reţele formate dinamic de un sistem autonom de noduri mobile, conectate prin intermediul undelor radio şi care nu utilizează o reţea existentă care să dispună de infrastructură. Reţelele MANET permit utilizatorilor să se conecteze în mod spontan cu alţi utilizatori din raza semnalului reţelei wireless [5]. Raza semnalului poate fi văzută precum o sferă digitală care înconjoară utilizatorul (Figura 3.5).

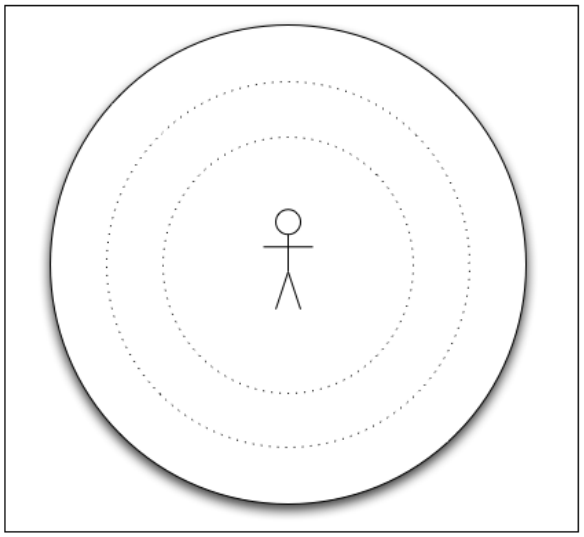

Figura 3.5 Sfera digitală din jurul utilizatorului

Reţelele MANET pot fi clasificate în funcţie de modul în care comunică. Astfel, putem vorbi de reţele ad-hoc singlehop şi multihop. În reţelele singlehop nodurile pot comunica în mod direct şi sunt unul în raza celuilalt (Figura 3.6). Versiunea actuală a framework-ului Peer2Me realizează o comunicare de acest gen.

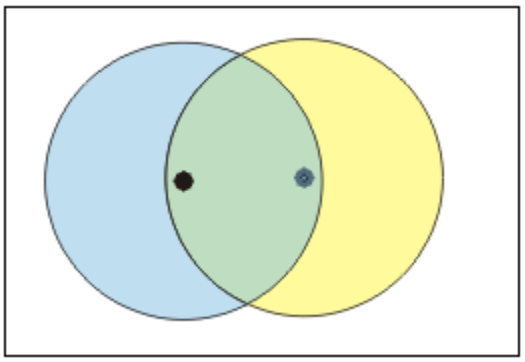

Figura 3.6 Reţea ad-hoc singlehop

Într-o reţea multihop, nu toate nodurile sunt conectate în mod direct datorită faptului că nu se găsesc în aceeaşi proximitate. Din această cauză comunicarea între aceste noduri va fi realizată cu ajutorul nodurilor intermediare. Figura 3.7 ilustrează o reţea ad-hoc multihop, comunicarea între cele mai îndepartate noduri fiind reprezentată printr-o linia neagră. Tema lucrării de diplomă constă tocmai în implementarea unui algoritm de rutare avansat necesar realizării unei reţele ad-hoc multihop.

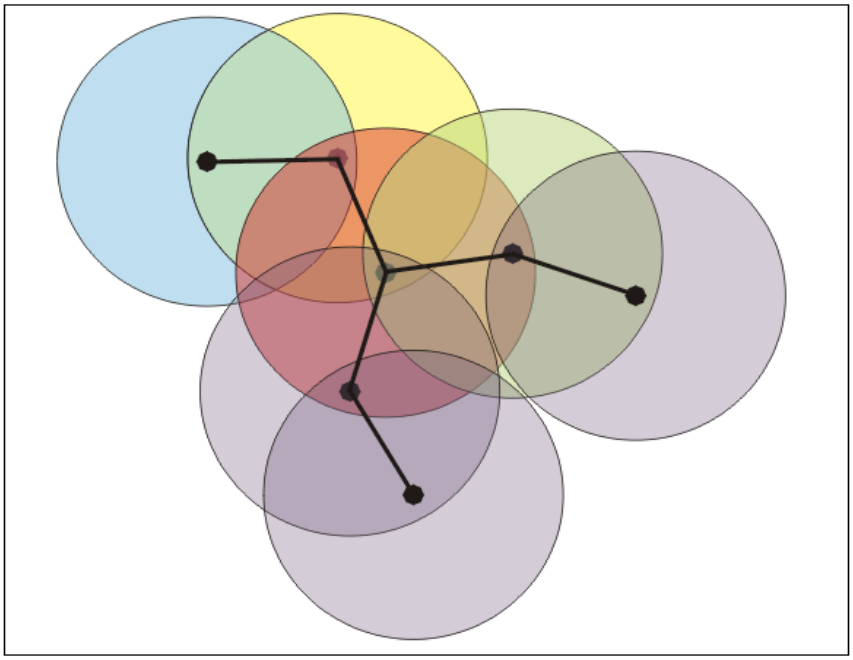

Figura 3.7 Reţea ad-hoc multihop

Lipsa unei infrastructuri cablate şi crearea rapidă a reţelei fac din acesta un candidat perfect pentru situaţiile de urgenţă, precum catastrofele naturale sau provocate de om, conflictele militare, situaţiile medicale de urgenţă ş.a.

Tehnologiile de reţea care suportă modelul peer-to-peer se încadrează în categoria reţelelor Wireless Personal Area Network (WPAN) prezentată anterior în cadrul acestui capitol. Exemple de astfel de tehnologii sunt Bluetooth, ZigBee şi IrDA.

3.4 Computer Suported Cooperative Work (CSCW)

În situaţiile în care realizarea unei activităţi necesită implicarea unui grup de indivizi apar probleme legate de comunicare şi sincronizare. Acest efort poate fi privit ca un efort comun depus pentru realizarea unui scop comun sau partajarea unei experienţe.

Domeniul CSCW se axează pe utilizarea calculatoarelor în sprijinul cooperării şi comunicării în efortul colaborativ. Acest domeniu poate fi reprezentat prin intermediul a două dimensiuni, timpul şi locaţia, precum ilustrat în figura 3.8.

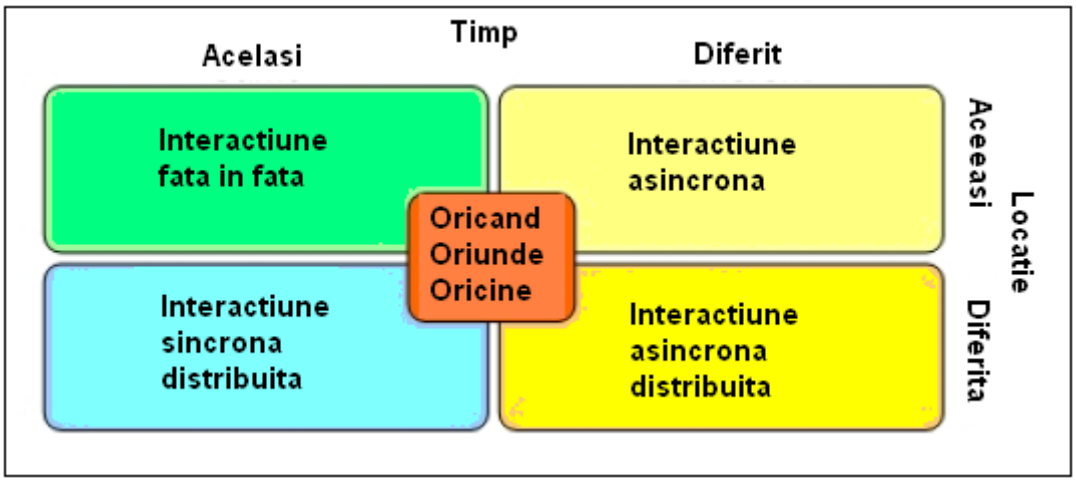

Figura 3.8 Dimensiunile CSCW

Cele mai multe probleme nerezolvate sunt legate de aplicaţiile care se încadrează în categoria „locaţie diferită”. Framework-ul Peer2Me este un framework utilizat în dezvoltarea de aplicaţii colaborative destinate telefoanelor mobile, aceste aplicaţii încadrându-se în categoriile „Interacţiune faţă în faţă” şi „Interacţiune asincronă”.

Capitolul 4. Tehnologii pentru medii mobile

Există o mare diferenţă între calculatoarele desktop şi telefoanele mobile datorată resurselor limitate de care dispun ultimele dintre acestea, motiv pentru care software-ului folosit în cazul calculatoarele desktop nu este adecvat dezvoltării de aplicaţii destinate telefoanelor mobile . În aceste condiţii platforma J2ME a fost considerată a fi cea mai adecvată soluţie în dezvoltarea de aplicaţii destinate telefoanelor mobile, software-ul fiind dedicat dispozitivelor cu resurse limitate.

4.1 Java 2 Micro Edition (J2ME)

Acest subcapitol oferă o scurtă introducere în platforma Java 2 Micro Edition, concentrându-se pe capabilităţile, posibilităţile şi limitările de care aceasta dispune în realizarea de aplicaţii pentru reţele de tipul peer-to-peer.

4.1.1 Arhitectura J2ME

Java 2 Micro Edition este un subset al platformei Java 2, aceasta fiind o platformă proiectată pentru dispozitive mici, dedicată dispozitivelor mobile de genul PDA-urilor şi a telefoanelor mobile. Acest fapt presupune existenţa unor limitari stricte de memorie, putere de procesare şi capabilităţi de I/O. Pentru a asigura o soluţie compatibilă varietăţii de dispozitive prezente pe piaţă, J2ME a fost împărţită în *Configuraţii*, *profile* şi *pachete opţionale***.** Arhitectura J2ME este prezentată în Figura 4.1 .

Framework-ul Peer2Me a fost construit având la bază arhitectura pentru dispozitive mobile cu resurse limitate, motiv pentru care în continuare accentul se va pune pe această arhitectură.

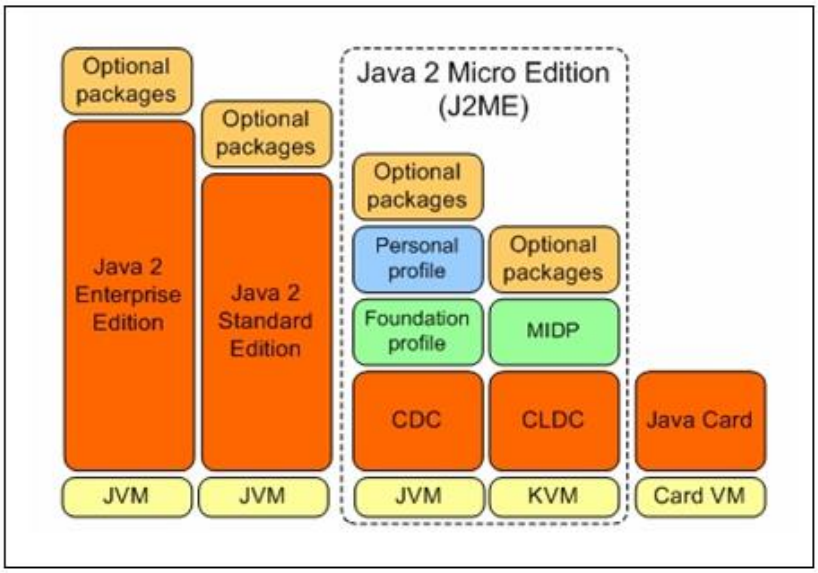

Figura 4.1 Arhitectura J2ME

Nivelul de Configuraţii

Nivelul de configuraţii reprezintă primul nivel din cadrul arhitecturii J2ME. Acest nivel este alcătuit din două tipuri separate de configuraţii: Connected Device Configuration (CDC) şi Connected Limited Device Configuration (CLDC).

CDC a fost proiectat pentru dispozitive puternice percum PDA-urile şi dispozitive sofisticate precum sistemele de navigaţie pentru maşini. Aceste dispozitive au procesoare pe 32 de biţi, cel puţin 2MB de memorie principală , 2.5 MB de ROM, şi anumite tipuri de conectivitate la internet. CDC foloseste maşina virtuala java (JVM) cu capacităţi maxime pentru J2SE.

CDLC este o configuraţie minimală pentru dispozitive cu multe constrângeri în ceea ce priveşte puterea de calcul, durata de viaţă a bateriei, memoria şi lăţimea de bandă. Specificaţiile CLDC 1.1 presupun următoarele necesităţi tehnice şi caracteristici ale unui dispozitiv:

- cel puţin 160 kb de memorie disponibilă pentru platforma Java;
- viteza procesorului începand de la 8-32 MHz;
- procesorul pe 16 sau 32 de biţi;
- putere limitată;
- conectivitate la anumite tipuri de reţele;
- volum mare de fabricare.

Aceast tip include maşina virtuală java a arhitecturii, numită Kilobyte Virtual Machine (KVM) şi un set de clase derivate din J2SE. Toate aplicaţiile care rulează pe J2ME sunt executate de către KVM.

Profile

Pentru a defini un ciclu de viaţă, o interfaţă utilizator şi un mod de acces la dispozitivul mobil este nevoie de un *profil*. Pentru configuraţia CLDC, firma Sun a definit un profil numit Mobile Information Device Profile (MIDP). Profilul MIDP defineşte un set de funcţionalităţi disponibile şi poate fi extins cu pachete opţionale.

Pachete optionale

În cadrul framework-ului Peer2Me este utilizat următorul pachet opţional:

- JSR82 75 – API-ul java pentru a accesa datele PIM-ul (Personal Information Management) şi sistemul de fişiere.

4.2 Tehnologia Bluetooth

Acestă secţiune oferă o scurtă introducere în tehnologia Bluetooth, aceasta fiind în momentul actual singura tehnologia de reţea implementată în framework-ul Peer2Me.

Bluetooth este o tehnologie cu cost redus, cu consum redus de energie şi rază de acţiune mică care are scopul de a înlocui conexiunile prin cablu dintre dispozitive precum PDA-uri, telefoane mobile sau alte dispozitive mobile. Tehnologia a fost iniţiată în anul 1994 când compania suedeză Ericson Mobile Comunication a început să cerceteze fezabilitatea unei tehnologii cu cost scăzut, consum scăzut de energie şi capabilă să comunice prin unde radio. Ericson a realizat faptul că pentru ca tehnologia să poată fi folosită este necesar ca ea să fie implementată pe cât mai multe dispozitive mobile, astfel încât în 1997 se decid să ofere tehnologia gratis. Doar un an mai târziu cinci companii: Ericson, IBM, Intel, Nokia şi Toshiba şi-au anunţat intenţia de a dezvolta această tehnologie. Noua tehnologie a primit numele de Bluetooth, iar cele cinci companii au creat grupul „Bluetooth Special Interest Group” (SIG) responsabil cu dezvoltarea tehnologiei [7]. Astăzi grupul SIG cuprinde următoarele opt companii: Agere, Ericson, IBM, Intel, Microsoft, Motorola, Nokia si Toshiba.

Tehnologia a fost numită astfel după numele unui rege viking danez care se numea Harald Bluetooth şi care a unit Norvegia cu Danemarca. Acest nume a fost ales în speranţa că tehnologia Bluetooth va reuni tehnologi diferite precum calculatoarele şi telefoanle mobile.

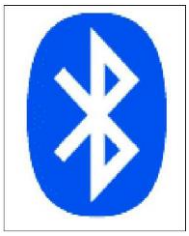

Figura 4.2 Logo Bluetooth

4.2.1 Comunicarea prin unde radio

O undă radio este un puls de energie electromagnetică. Undele radio sunt transmise atunci când un transmiţător oscilează la o frecvenţă specifică, şi, cu cât oscilează mai repede, cu atât frecvenţa de transmitere este mai mare. Pentru amplificarea şi transmiterea undei radio se foloseşte o antenă, iar pentru primirea unui semnal radio se utilizează un receptor radio. Datorită faptului că se utilizează frecvenţe radio diferite pentru diferite tipuri de comunicaţii, receptorul trebuie setat să „asculte” pe o anumită frecvenţă. Tehnologia Bluetooth operează pe banda de 2.4 GHz, aceata fiind o bandă de frecvenţe gratuită.

4.2.2 Rata de transfer

În teorie, implementarea cu Bluetooth 1.0 ar trebui sa atingă viteza maximă de 1Mbps. Bluetooth-ul suportă atât transmisii simetrice cât şi transmisii asimetrice. În cazul transmisilor simetrice (aceeaşi viteză în ambele sensuri), viteza maximă este de 432.6Kbps. În cazul transmisilor asimetrice (viteză ridicată într-o direcţie, viteză mică în cealaltă direcţie), viteza maximă este de 721Kbps la ieşire şi de 56Kbps la intrare. Aceste viteze sunt valabile numai în cazul transmisilor de date. Semnalele de voce sunt transmise cu o rată maximă de 64Kbps în ambele direcţii [7].

Specificaţiile pentru Bluetooth 2.0 sunt compatibile cu cele din versiunea 1.0. Marea diferenţă o reprezintă introducerea lui EDR (Enhanced Data Rate) de 2.1 Mbps. Tehnic dispozitivele care suportă versiunea 2.0 au un consum de energie mai ridicat, dar rata de trasnmisie de trei ori mai mare reduce timpul de trasnmisie, reducând astfel consumul de energie la jumătate din consumul dispozitivelor care dispun de versiunea 1.0.

4.2.3 Securitatea Blutooth-ului

Tehnologia Bluetooth utilizează algoritmul SAFER+ [8] pentru autentificarea şi genererea de chei. Pentru criptarea pachetelor se utilizează E0 [9].

4.2.4 Piconet si Scatternet

În momentul în care două sau mai multe dispozitive Bluetooth stabilesc o conexiune directă are loc crearea unui tip special de PAN numit piconet. Teorectic, un piconet Bluetooth poate conţine cel mult opt dispozitive, restul sunt sclavi (Figura 4.3). În timp ce un dispozitiv se comportă precum un nod şef, celelalte noduri se comportă precum muncitori [7] .

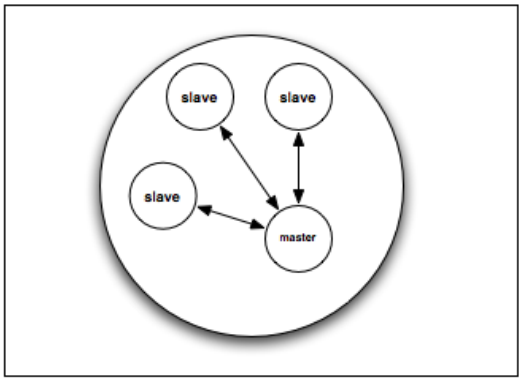

Figura 4.3 Un piconet cu patru noduri

Un dispozitiv dintr-un piconet poate comunica de asemenea cu dispozitive dintr-un alt piconet. Astfel, din mai multe reţele piconet se crează o reţea scatternet (Figura 4.4). Pentru a realiza comunicarea dintre piconet-urile ce alcătuiesc un scatternet, unele noduri sunt nevoite să înainteze pachetele între piconet-uri în numele celorlalte noduri. Acest proces necesită implementarea unui algoritm de rutare avansat. Studiul şi implementarea unui astfel de algoritm a constituit tema acestei lucrării de diplomă.

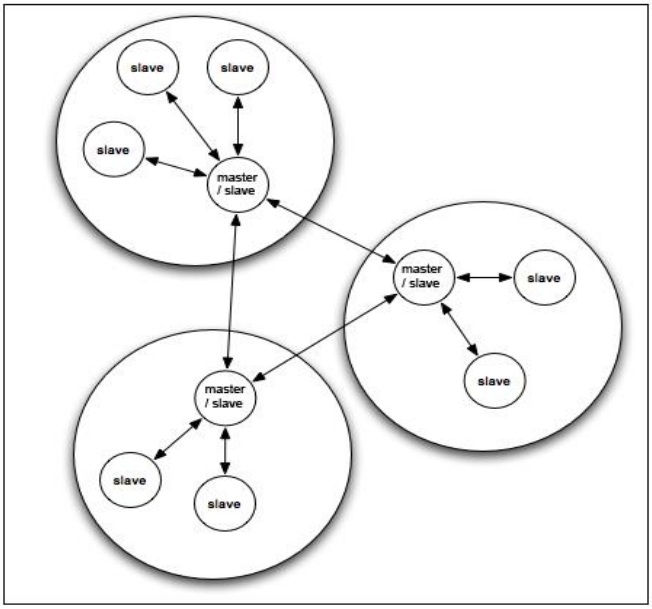

Figura 4.4 Un scatternet alcătuit din trei piconet-uri

Partea III

Framework-ul Peer2Me

Capitolul 5. Framework-ul Peer2Me

Ca rezultat al studiului bibliografic efectuat, framework-ului Peer2Me a fost considerat candidatul ideal în realizarea acestei lucrării de licenţă. Acesta permite dezvoltarea de aplicaţii colaborative pentru telefoane mobile care creează o reţea de tipul Personal Area Network (PAN). Prima versiune a framework-ului este rezultatul lucrării de master [11] dezvoltată la Universitatea de Ştiinţă şi Tehnologie din Norvegia, oferind funcţionalitate specifică reţelelor peer-to-peer hibride. Cea de-a doua versiune a constat în optimizarea framework-ului prin realizarea unei reţele peer-to-peer pure.

5.1 Caracteristici

În acest subcapitol sunt descrise principalele caracteristici funcţionale şi non-funcţionale ale framewok-ului Peer2Me.

5.1.1 Caracteristici funcţionale

Caracteristicile funcţionale ale framework-ului Peer2Me sunt descrise în tabelul 5.1.

Caracteristici Funcţionale	Descrierea caracteristicilor
CF1	Framework-ul suportă telefoanele mobile.
CF2	Framework-ul suportă dezvoltarea de reţele ad-hoc.
CF3	Framework-ul este capabil să se conecteze la o reţea ad-hoc existentă.
CF4	Nodurile din reţea sunt capabile să schimbe mesaje între ele.
CF5	Framework-ul este capabil să creeze un grup de noduri legate de o aplicaţie specifică.
CF6	Framework-ul suportă trimiterea mesajelor multicast şi broadcast în interiorul grupului.
CF7	Framework-ul este capabil să descopere alte telefoane mobile care utilizează aceleaşi servicii.
CF8	Framework-ul suportă crearea unor grupuri deschise sau închise.
CF9	Framework-ul permite unui nod să încerce să intre într-un grup închis.
CF10	Framework-ul permite nodurilor dintr-un grup închis să respingă alte noduri care încearcă să se conecteze la grup.

CF11	Framework-ul permite unui nod să intre într-un grup deschis.
CF12	Framework-ul este capabil să prezinte mesaje de decizie unui utilizator.
CF13	Framework-ul este capabil să prezinte informaţii utilizatorului despre evenimentele apărute.
CF14	Framework-ul suportă diferite medii de reţea.
CF15	Aplicaţiile sunt independente indiferent de mediul reţelei folosit în acel moment în interiorul sistemului. Aplicaţiile care folosesc sistemul, pentru a se ocupa de traficului din reţea, nu trebuie să cunoască mediu de reţea folosit de dispozitiv .
CF16	Framework-ul este capabil să identifice de unde a fost iniţiat un transfer pentru a fi capabil să trimită un răspuns acelui dispozitiv.

Tabelul 5.1 Caracteristici funcţionale

Aceste caracteristici reprezintă cerinţele funcţionale care stau la baza versiunii 2.0 a framework-ului, cerinţe cărora li se va adăuga urmatoarea cerinţă funcţională, impusa de algoritmul implementat

CF17	**Framework-ul trebuie sa fie capabil de rutarea mesajelor între noduri. Un nod trebuie să poată trimite un mesaj unui alt nod identificat prin numărul de telefon.**

Tabelul 5.2 Noua cerinţă funcţională

Diagrama generală a Cazurilor de utilizare a sistemului rezultată din caracteristicile funcţionale este prezentată în figura 5.1. Principalele cazurile de utilizare ale framework-ului constau în:

- stabilirea unei conexiuni între utilizatori;
- crearea unui grup rezultat în urma conexiunii nodurilor dintr-o reţea ad-hoc;
- sincronizarea necesară între nodurile din interiorul unui grup;
- schimbul de date între noduri;
- notificarea MIDlet-ului cu privire la evenimentele care au avut loc, spre exemplu primirea unui nou mesaj, adăugarea unui nou nod la reţea;
- selectarea mediului reţelei, momentan fiind disponibil doar mediul Bluetooth;
- utilizarea log-ului definit în cadrul framework-ului.

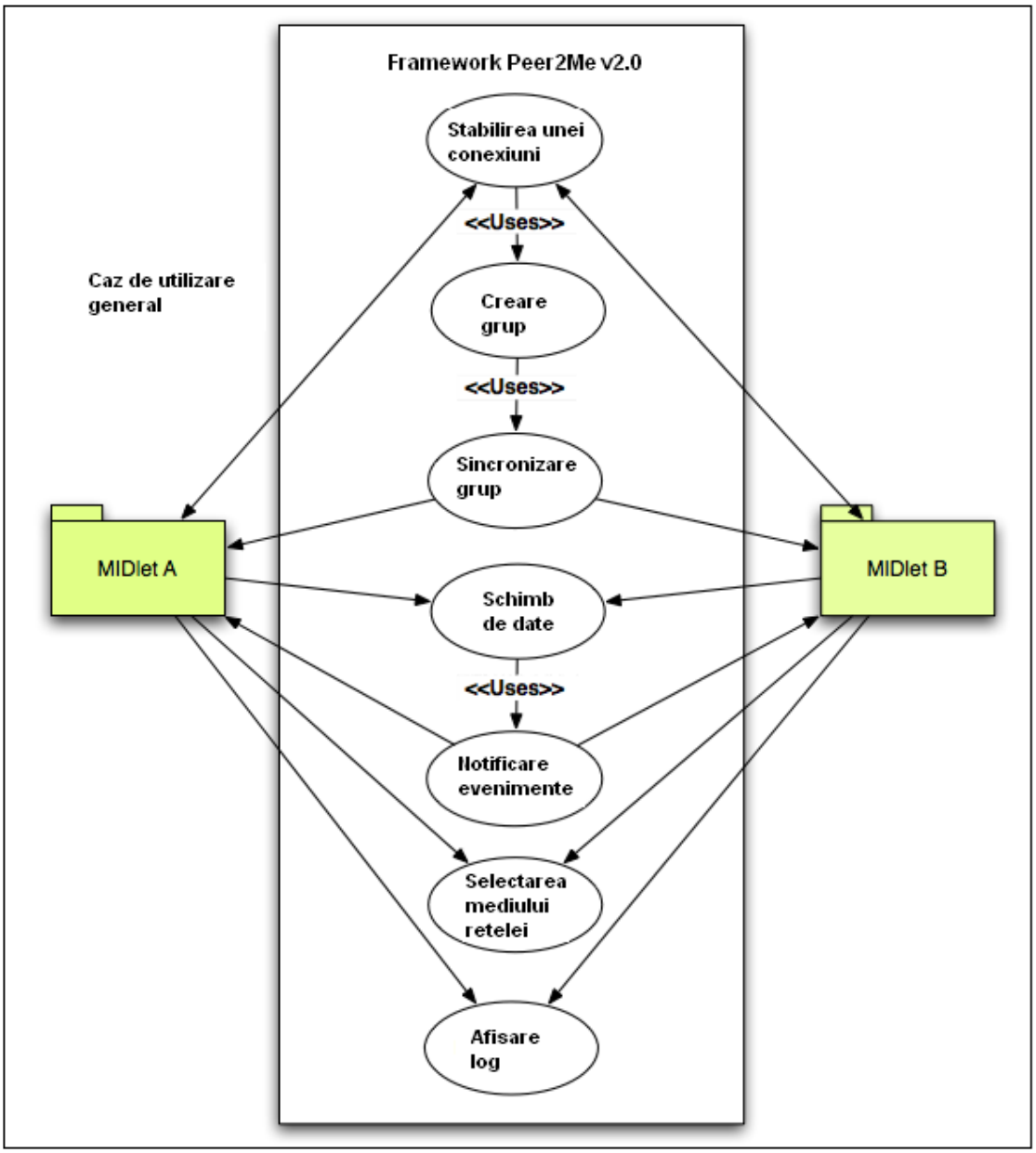

Figura 5.1 Caz de utilizare general al Framewok-ului Peer2Me

În continuare va fi prezentat cazul de utilizare corespunzător schimbului de date dintre cele două MIDlet-uri. Alegerea acestui caz de utilizare se datorează faptului că lucrarea de diplomă adresează tocmai acest caz.

Caz de utilizare: Schimb de date
ID: UC4
Actori: MIDlet A...n

Precondiţii: 1. Framework-ul Peer2Me trebuie să fie iniţializat pe un telefon mobil cu suport pentru reţea. 2. Trebuie să existe o conexiune între două sau mai multe dispozitive. 3. Trebuie să existe un grup cu doi sau mai mulţi participanţi. 4. Grupul trebuie să fie sincronizat.
Desfăşurarea evenimentelor: 1. MIDlet-ul A vrea să trimită date prin intermediul framework-ului. 2. Framework-ul creează un pachet de date corespunzător, pe care îl trimite apoi fie prin intermediul algoritmului de rutare, fie în mod direct, la recipienţi. 3. MIDlet-ul n primeşte datele prin intermediul unei notificări a evenimentului.
Postcondiţii: 1. Are loc schimbul de date.
Desfaşurarea alternativă a evenimentelor: 1. MIDlet-ul A vrea să trimită date în reţea prin intermediul framework-ului. 2. Framework-ul creează un pachet de date corespunzător, pe care îl trimite apoi fie prin intermediul algoritmului de rutare, fie în nod direct, la recipienţi. 3. Datele nu au putut ajunge la recipienţi.
Postcondiţii: 1. Nu are loc schimbul de date.

5.1.2 Caracteristici non-funcţionale

Caracteristicile non-funcţionale sunt caracteristici cu un caracter puţin mai neclar decât cel al caracteristicilor funcţionale descrise în subcapitolul anterior. De obicei nu sunt foarte clar specificate de către utilizatorii sistemului, dar asta nu înseamnă că sunt mai puţin importante pentru satisfacerea necesităţii utilizatorilor şi pentru arhitectura sistemului. Aceste caracteristici nu pot fi reprezentate prin intermediul Cazurilor de Utilizare. În tabelul 5.3 sunt prezentate caracteristicile non-funcţionale ale framework-ului Peer2Me.

Caracteristici Non-funcţionale	Descrierea caracteristicilor
CNF1	Framework-ul este capabil să transfere mesaje suficient de rapid pentru

	interacţiunea în timp real. Acest lucru înseamnă că mesajele de lungime normală trebuie să dea impresia de apariţie instantanee pe telefonul remote.
CNF2	Framework-ul este capabil să detecteze deconectarea nodurilor de la grup şi să notifice aplicaţiile şi nodurile cu privire la aceste evenimente.
CNF3	Framework-ul reuşeşte să se adapteze la eroriile care apar datorită naturii instabile a reţelelor wireless.
CNF4	Framework-ul împiedică primirea mesajelor de către alte aplicaţii, înafara celor adresate.

Tabelul 5.3 Cerinţe Non-funcţionale

Aceste caracteristici non-funcţionale stau la baza următoarelor criterii de analiză a calităţii framework-ului: gradul de utilizare, performanţa, costurile aduse de schimbările survenite la nivel de framework, disponibilitatea, securitatea şi nivelul de testare a acestuia.

5.1.3 Cerinţe de sistem

În acest capitol este descris pe scurt mediul necesar aplicaţiei pentru a rula într-un mod corespunzător.

J2ME – MIDP 2.0 – Atât framework-ul cât şi aplicaţiile dezvoltate sunt implementate în Java şi de aceea depind de suportul J2ME şi MIDP 2.0.

Sistemul de operare - Atâta timp cât telefonul mobil pe care se rulează aplicaţia dispune de tehnologiile specificate anterior, sistemul de operare folosit este irelevant. Acest fapt se datorează independenţei platformei limbajului de operare Java.

Memoria – Memoria necesară variază în funcţie de numărul de aplicaţii inclus în fişierul JAR.

Afişajul – Interfaţa Utilizator a aplicaţiilor trebuie să fie operaţională şi funcţională pe orice tip de dispozitiv mobil atâta timp cât acesta îndeplineşte condiţiile specificate mai sus.

Bluetooth – Dispozitivul mobil trebuie sa aibă suport pentru Bluetooth, acesta fiind singurul mediu disponibil momentan în framework-ul Peer2Me. În plus, API-ul JSR-82 pentru J2ME trebuie sa fie disponibil, pentru a permite aplicaţiilor să acceseze funcţionalităţiile tehnologiei Bluetooth.

5.2 Arhitectura Framework-ului Peer2Me

Acest subcapitol se axează pe prezentarea arhitecturii framework-ului Peer2ME, o arhitectură strict bazată pe module, cu pachete separate pentru fiecare tip major de clase (Figura 5.2). Toate pachetele sunt sub module ale sistemului, iar pachetul bluetoothNetwork este un sub modul al pachetului network.

Una dintre cele mai importante caracteristici ale acestei arhitecturi este faptul că tot codul legat de tehnologia Bluetooth este localizat într-un sub modul al pachetului network, ceea ce înseamnă că restul codului este în totalitate independent de reţea. În acest mod o conversie ulterioară a framework-ului la o tehnologie alternativă nu va afecta clasele framework-ului care nu depind de reţea. Este posibilă încorporarea unei noi tehnologi de reţea direct în sistemul curent fără nici o schimbare.

În continuare sunt prezentate pe scurt pachetele care alcatuiesc framework-ul Peer2Me v2.0.

Framework – Acest pachet conţine clasa FrameworkFrontEnd care este „inima” acestui framework-ului sunt realizate prin intermediul acestei clase. Două interfeţe conectează MIDlet-ul cu clasa FrameworkFrontEnd. Datorită acestui fapt restul framework-ului este „ascuns” eventualilor dezvoltatori. sistem. Toate comunicaţiile dintre interfaţa utilizator a MIDlet-ului şi funcţionalitatea

Domain – Pachetul conţine clasele bazate pe domeniul conceptual al framework-ului precum clasa Group, Node, DataPackage şi altele.

Network – Acest pachet conţine clasele referitoare la comunicaţile din reţea. Clasa ConnectionListener tratează conexiunile iniţiate de alte dispozitive, iar clasa NodeConnection reprezintă o conexiune la fiecare nod remote din interiorul aceluiaşi grup.

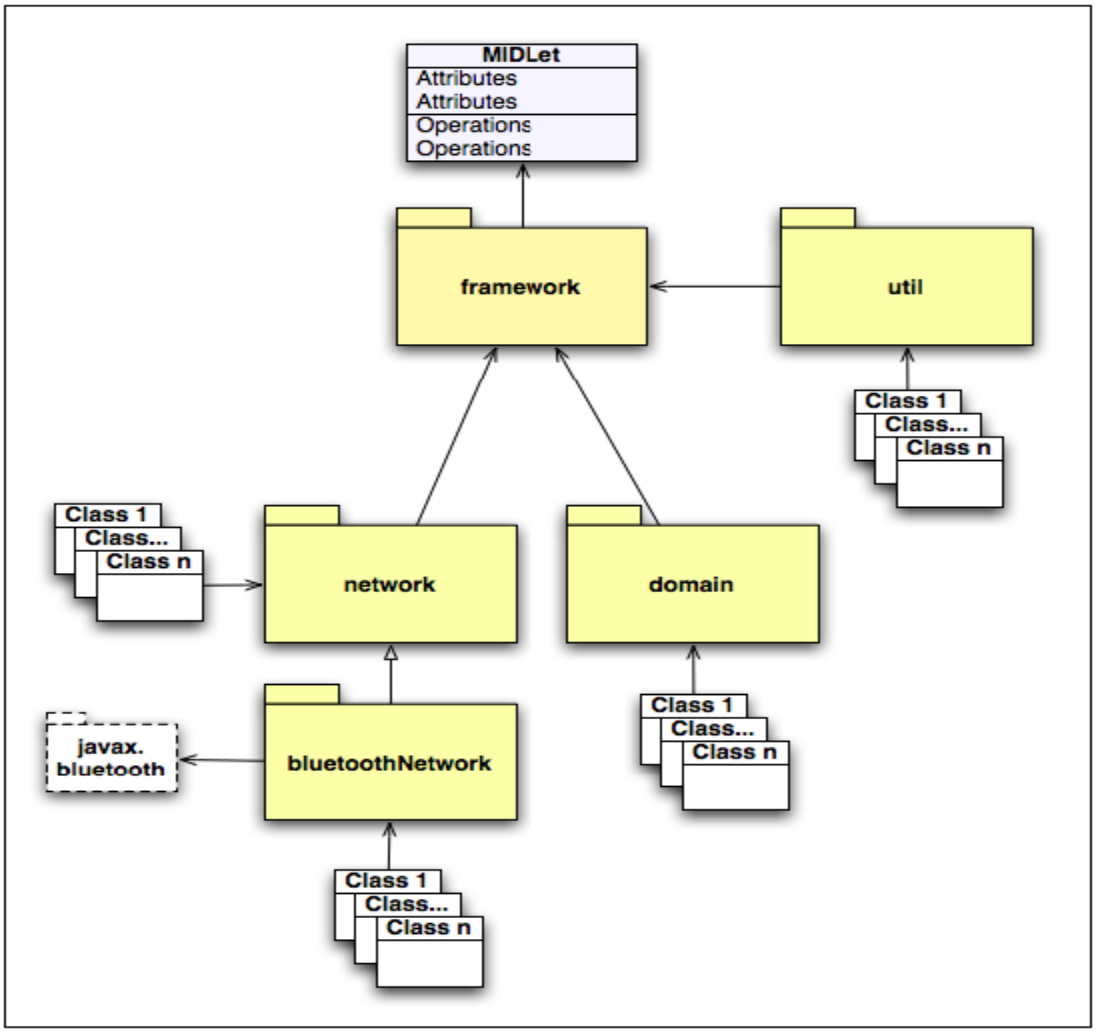

Figura 5.2 Arhitectura Framework-ului Peer2Me

BluetoothNetwork – Acest pachet este responsabil pentru toate operaţiile de reţea specifice tehnologiei Bluetooth. Prin înlocuirea acestui pachet cu un alt modul network, frameworkul ar putea comunica de exemplu într-o reţea WLAN.

Util – Clasele din pachetul Util conţin functionalităţi utile folosite de alte clase din framework. Aceste clase pot fi folosite din orice altă clasă în orice pachet al framework-ului.

MIDlets – Reprezintă MIDlet-urile care rulează pe framework. Acestea conţin funcţionalitatea şi interfaţa utilizator unică pentru fiecare aplicaţie. MIDlet-urile pot folosi avantajele framework-ului Peer2ME prin intermediul interfeţei framework-ului.

5.2.1 Arhitectura detaliată a framework-ului Peer2Me

În acest subcapitol vor fi descrise principalele modificări aduse arhitecturii framework-ului Peer2Me, modificări ilustrate prin intermediul diagramelor de clasă corespunzătoare fiecărui pachet din figura 5.2.

5.2.1.1 Pachetul framework

Modificările din clasa FrameworkFrontEnd sunt reprezentate de implementarea a două noi metode numite *sendSmsPackage* şi *notifyAboutReceivedSmsPackage*. Cele două metode realizează trimiterea unui pachet de tipul *SmsPackage*, respectiv, notifică utilizatorul de primirea unui nou *SmsPackage*. Metodele au fost creeate după exemplu oferit de metodele *sendTextPackage* şi *notifyAboutReceivedTextPackage*, conferind astfel omogenitate în implementarea framework-ului.

Metoda *sendSmsPackage* creează un pachet de tipul *SmsPackage* pe baza informaţiilor primite de la MIDlet, iar apoi îl trimite în reţea utilizând mediul de reţea disponibil, la momentul actual acesta fiind realizat prin tehnologia Bluetooth.

Metoda *notifyAboutReceivedSmsPackage* înştinţează MIDlet-ul cu privire la primirea unui nou pachet de tipul *SmsPackage* şi tot odată adaugă această informaţie la log-ul framework-ului.

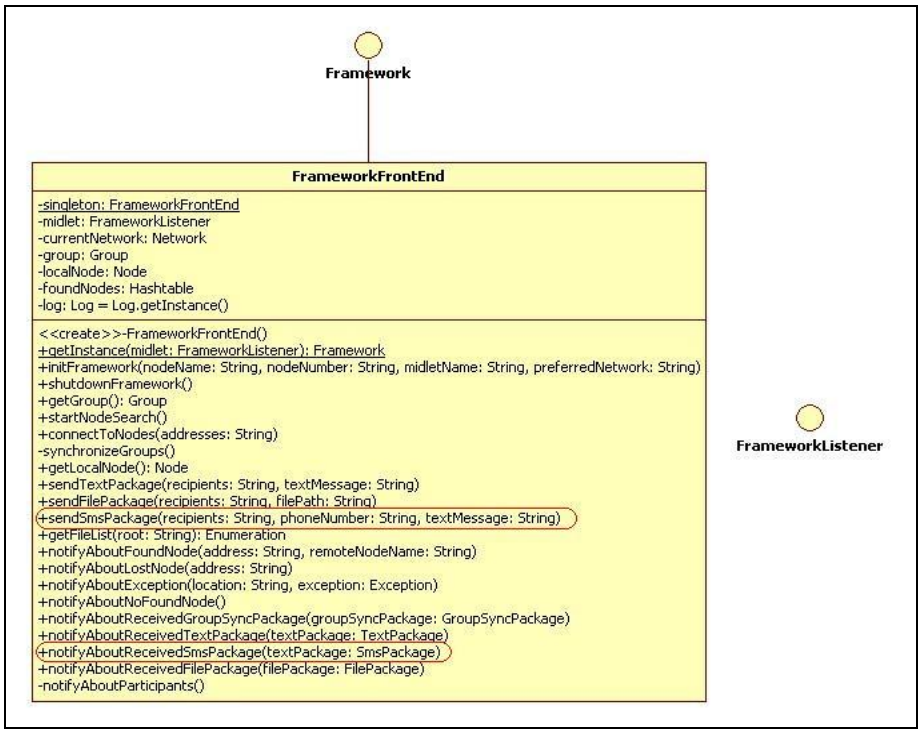

Figura 5.3 Pachetul framework

5.2.1.2 Pachetul domain

Pachetul domain conţine noile clase *SmsPackage* şi *RealisationPackage* necesare realizării procesului de trimitere a unui mesaj gen „sms”.

Clasa *SmsPackage* are declaraţi doi constructori utilizaţi pentru crearea unui nou pachet *SmsPackage* utilizând informaţiile primite de la MIDlet-ul local, respectiv, pentru creearea unui pachet gol de tipul *SmsPackage* care va fi umplut ulterior cu informaţiile extrase dintr-un pachet primit de la un nod remote, cu ajutorul metodei *parseByte*. Metodele implementate în clasa *SmsPackage* sunt următoarele:

- metoda *getContent* returnează conţinutul efectiv al mesajului;
- metoda *getCounter* returnează valoarea contorul mesajelor, contor utilizat pentru asocierea unui Id unic mesajului trimis;
- metoda *setRecipientFromSms* va seta valoarea recipientului extras din mesajul sms primit;
- metoda *getRecipientFromSms* returnează valoarea recipientului extras din mesajul sms primit;
- metoda *setClockFromSms* setează valoarea ceasului logic extras din mesajul sms primit;
- metoda *getClockFromSms* returnează valoare ceasului logic extras din mesajul sms primit;
- metoda *setCounterFromSms* setează valoarea contorului de mesaje extras din mesajul sms primit;
- metoda *getCounterFromSms* returnează valoarea contorului de mesaje extras din mesajul sms primit;
- metoda *toSendableFormat* creează pachetul care trebuie trimis în reţea după o anumită structură, după care are loc conversia pachetului într-un şir de biţi care va fi trimis în reţea; Această metodă este folosită în cazul în care procesul de creare a pachetului se datorează apelului din *sendSmsPackage*, în acest caz fiind necesară incrementarea contorului mesajelor, datorită faptului că MIDlet-ul local este chiar expeditorul mesajului;
- metoda *toSendableFormatForRouting* are ca idee acelaşi scop precum metoda anterioară, singura diferenţă dintre cele două metode constând în contorul mesajului folosit pentru creearea pachetului; Această metodă este folosită pentru

rutarea unui pachetului în rețea, caz în care contorul de mesaje nu va trebui incrementat;

- metoda *parseBytes* extrage datele din pachetul primit şi asignează aceste date anumitor atribute, necesare ulterior în creearea pachetului care trebuie rutat sau în funcţionarea algoritmului de rutare Scribble-M.

Clasa *RealisationPackage* este utilizată în etapa de finalizare a protocolului de rutare. În momentul în care pachetul a ajuns la nodul destinaţie se trimite un astfel de pachet în rețea. Pachetul va călătorii un singur hop, având ca obiectiv înştinţarea nodului de la care a fost primit pachetul, asupra faptului ca mesajul a ajuns la destinaţie, iar procesul expedierii periodice a mesajului trebuie anulat. Informaţia conţinută în acest pachet este reprezentată de Id-ul mesajului ajuns la nodul destinaţie.

Alte modificări care au survenit în acest pachet sunt reprezentate de introducerea a două noi tipuri de pachete în clasa *DataPackeage*, numite *SMS_PACKAGE* şi *REALISATION_PACKAGE*, dar şi de introducerea de noi metode şi atribute în clasa *Node* a pachetului.

În clasa *Node* au fost introduse două noi atribute, *nodeNumber* reprezentând numărul de telefon corespunzător unui nod, şi *smsCounter* reprezentând contorul mesajelor asociat nodului local. De asemenea, au fost modificaţi doi constructori, utilizaţi în crearea nodului local dar şi a nodurilor participante la grup, prin integrarea numărului de telefon corespunzător fiecărui nod din rețea. Metodele introduse sunt următoarele:

- metoda *getNodeNumber* returnează numarul de telefon al nodului;
- metoda *getSmsCounter* returnează numarul de mesaje trimise;
- metoda *setSmsCounter* setează numărul de mesaje trimise;
- metoda *restoreNodeFromSms* creează un nou nod utilizând informaţiile extrase dintr-un pachet *SmsPackage* primit de nodul local;

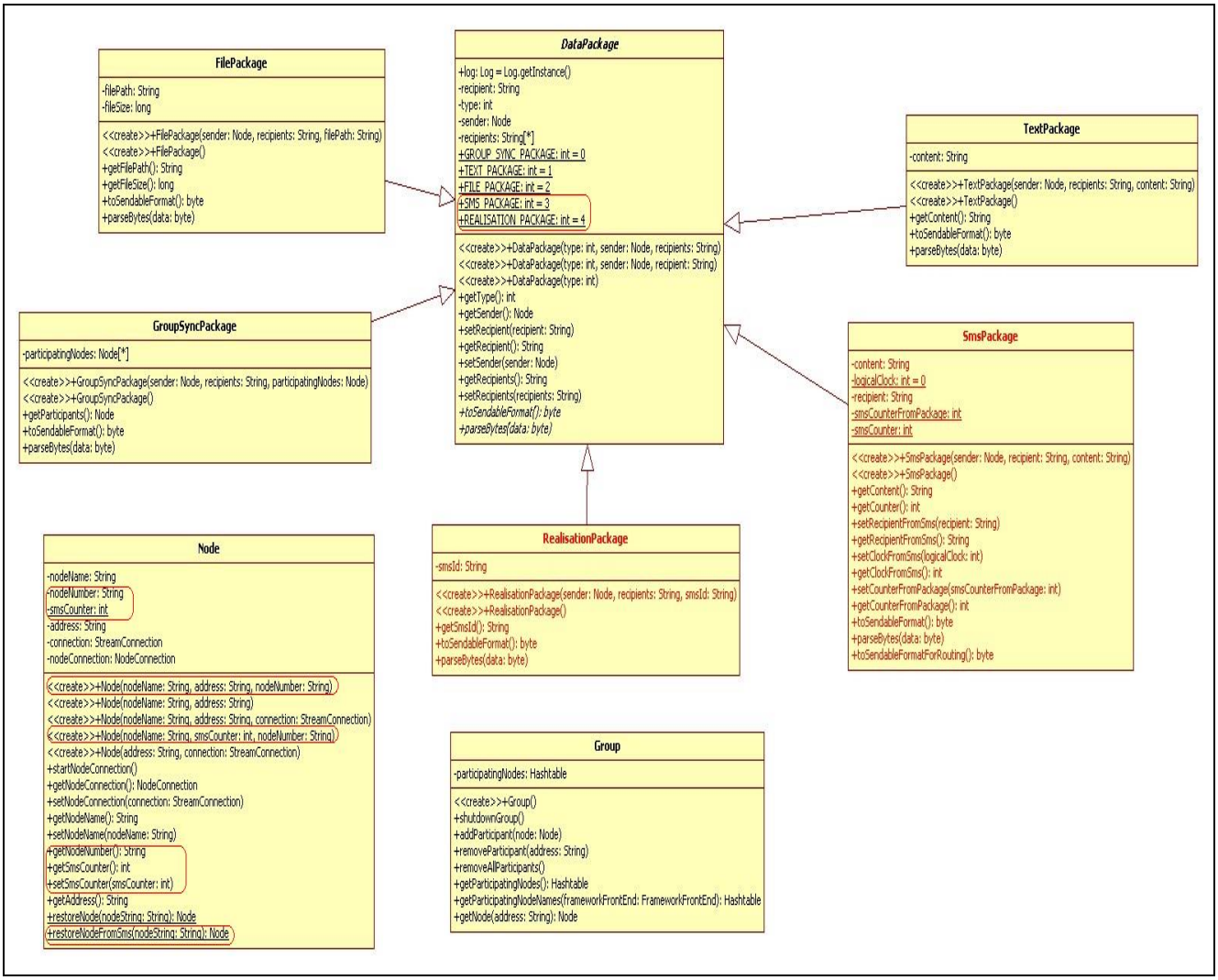

Figura 5.4 Pachetul domain

5.2.1.3 Pachetul network

Modificările aduse pachetului *network* sunt reprezentate de declararea a trei noi metode abstracte în clasa *Network* :

- metoda *scheduleSendDataPackage*;
- metoda *cancelSendDataPackage*;
- metoda *sendRealisationPackage*;

dar şi de introducerea logicii corespunzătoare algoritmului de rutare Scribble-M în clasa *NodeConnection*.

Pentru ca algoritmul să funcţioneze corespunzător, a fost creată o noua clasă privată *ScheduleThread* în clasa *NodeConnection*, clasă care extinde clasa *Thread*.

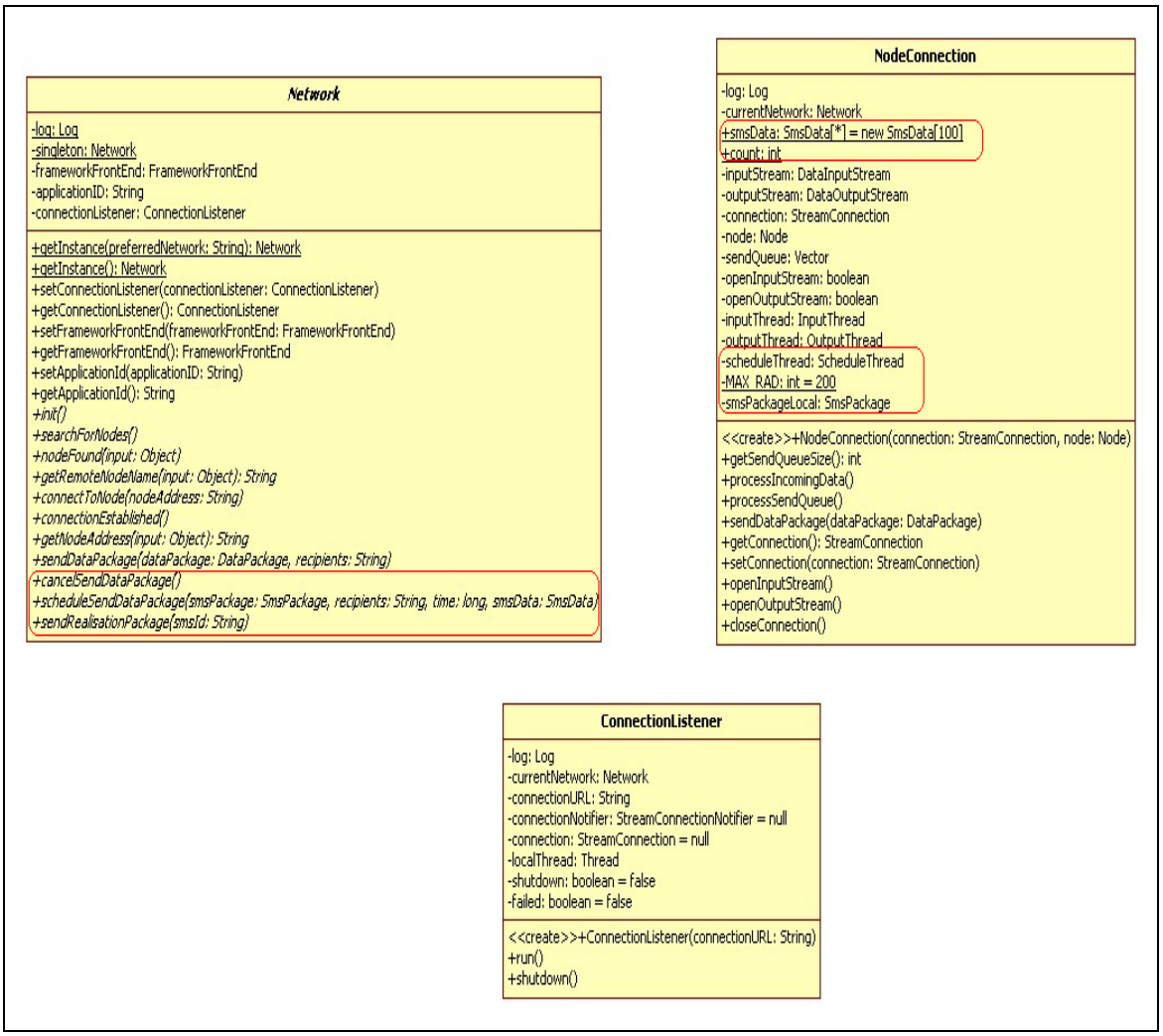

Figura 5.5 Pachetul network

5.2.1.4 Pachetul bluetoothNetwork

În cadrul acestui pachet, singurele modificări realizate au fost în clasa *BluetoothNetwork*, astfel încât a fost introdus un nou atribut, *cancelSendSmsPackege*, şi implementate cele trei metode noi declarate în clasa *Network*.

Atributul *cancelSendSmsPackage* este de tipul boolean, valoarea acestui atribut influenţând expedierea sau anularea expedierii unui pachet programat spre a fi trimis în reţea. Iniţial, atributul a fost setat cu valoarea *false*, pentru ca în momentul în care se primeşte un *RealisationPackage* sau ca urmare a îndeplinirii unor alte condiţii impuse de algoritmul de rutare, aceast atribut să primească valoarea *true*, prin intermediul noii metodei *cancelSendSmsPackage* care a fost creeată tocmai în acest scop.

Metoda *scheduleSendSmsPackage* programează trimiterea unui pachet de tipul *SmsPackage* dupa un anumit interval de timp primit ca parametru de intrare. Dacă după scurgerea acestui interval de timp, valoarea atributului *CancelSendSmsPackage* are valoarea *false* pachetul va fi trimis în rețea, în caz contar, procesul se termină ca urmare a primirii unui *RealisationPackage* de la nodul destinatar al mesajului, sau ca urmare a îndeplinirii unor alte condiţii impuse de algoritmul de rutare.

Metoda *sendRealisationPackage* realizează trimiterea unui pachet de tipul *RealisationPackage* la toţi participanţii grupului.

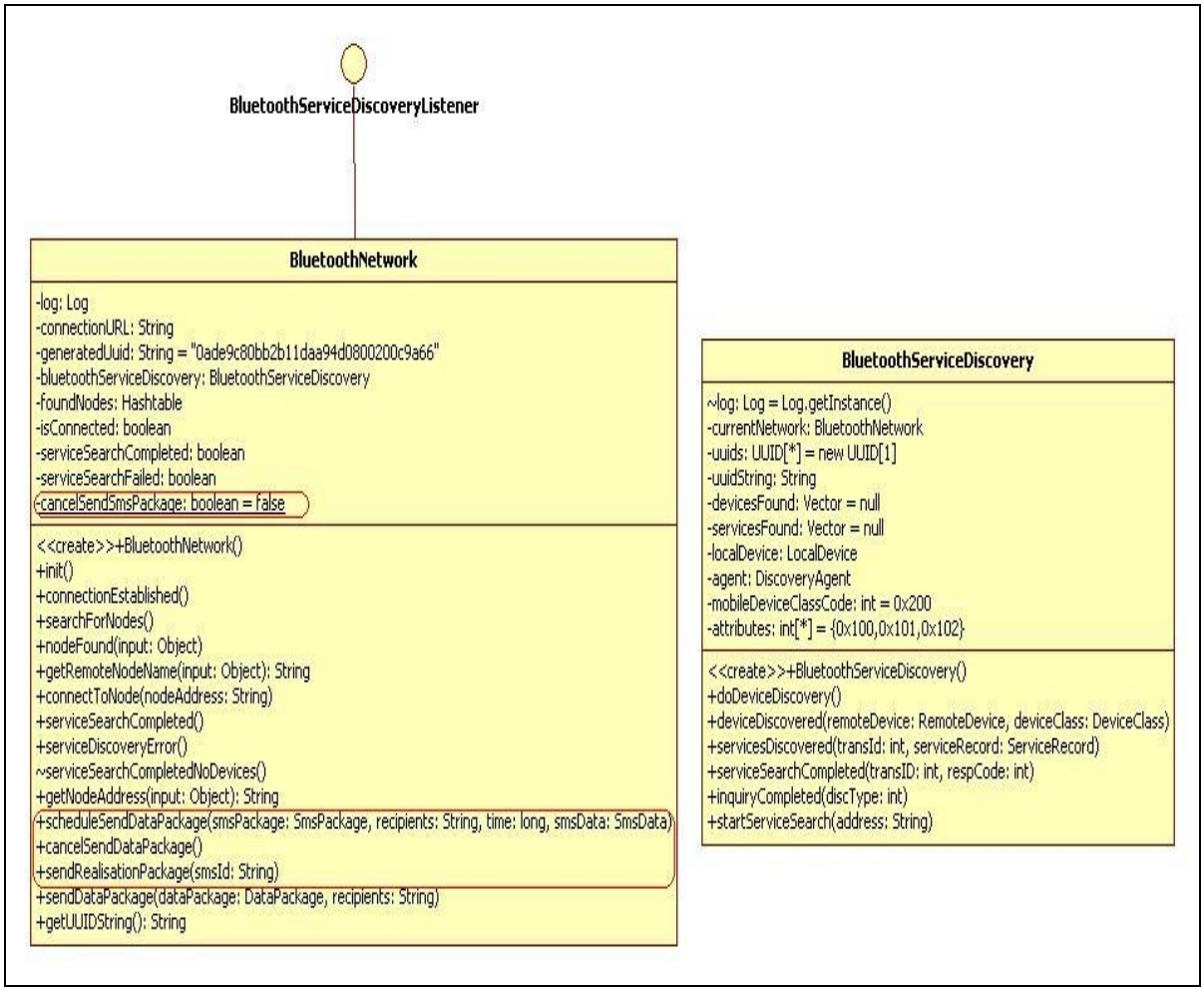

Figura 5.6 Pachetul bluetoothNetwork

5.2.1.5 Pachetul util

Noua clasă a pachetului util se numeste *SmsData*. Această clasă deţine un rol foarte important în realizarea algoritmului de rutare. Clasa defineşte şase atribute: *smsId-ul* (id-ul

mesajului), *logicalClock* (ceasul logic al mesajului), *lastReceived* (ultima dată cand a fost primit acest pachet), *realised* (dacă este *true*, atunci mesajul a ajuns la nodul destinaţie), *passiv* (dacă este *true*, nodul este passiv cu privire la nodul primit), responsable (dacă este *true*, nodul este responsabil cu privire la pachetul primit).

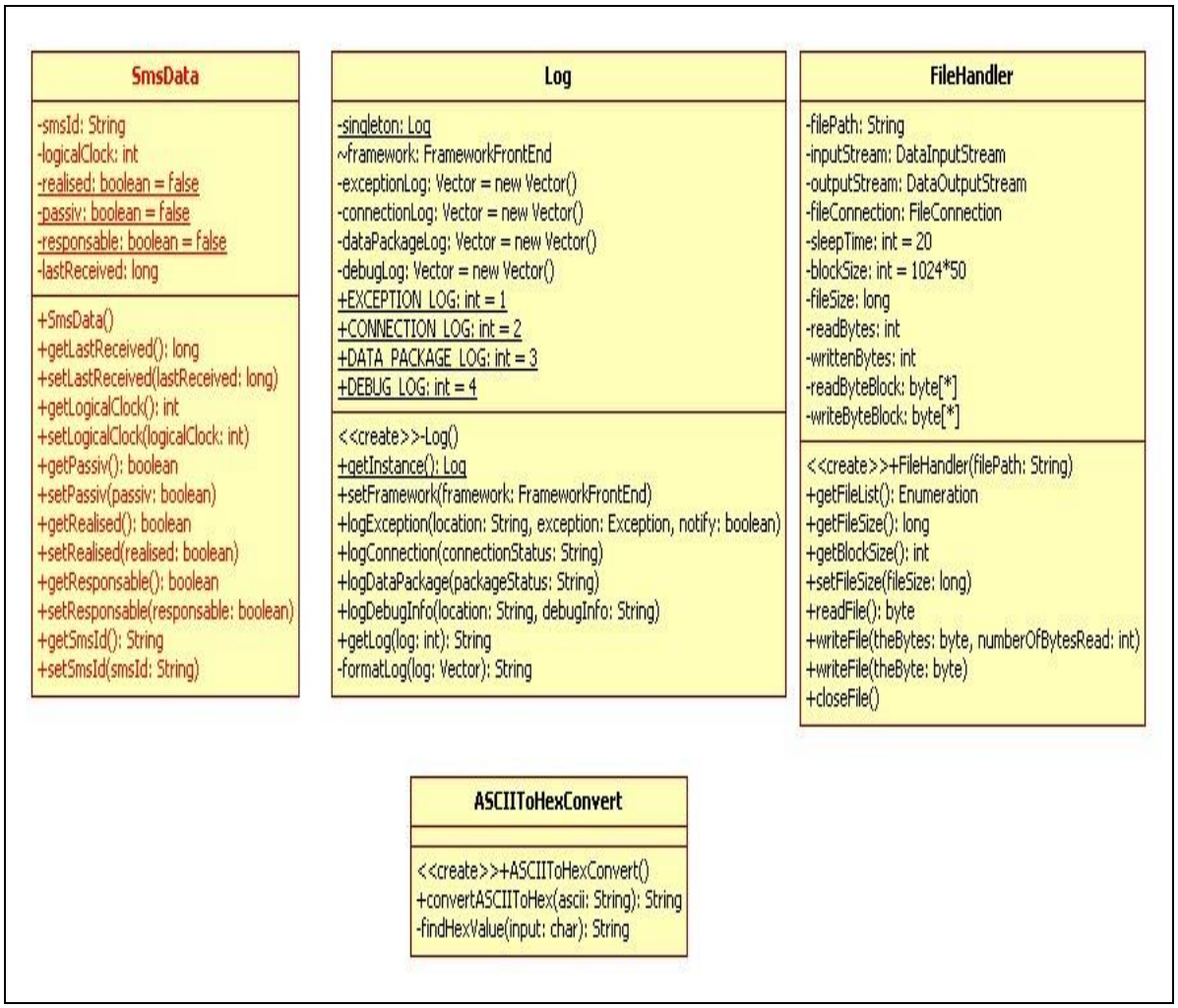

Figura 5.7 Pachetul util

Diagrama de clase generală aparţinând noului framework Peer2Me este prezentată în figura 5.8.

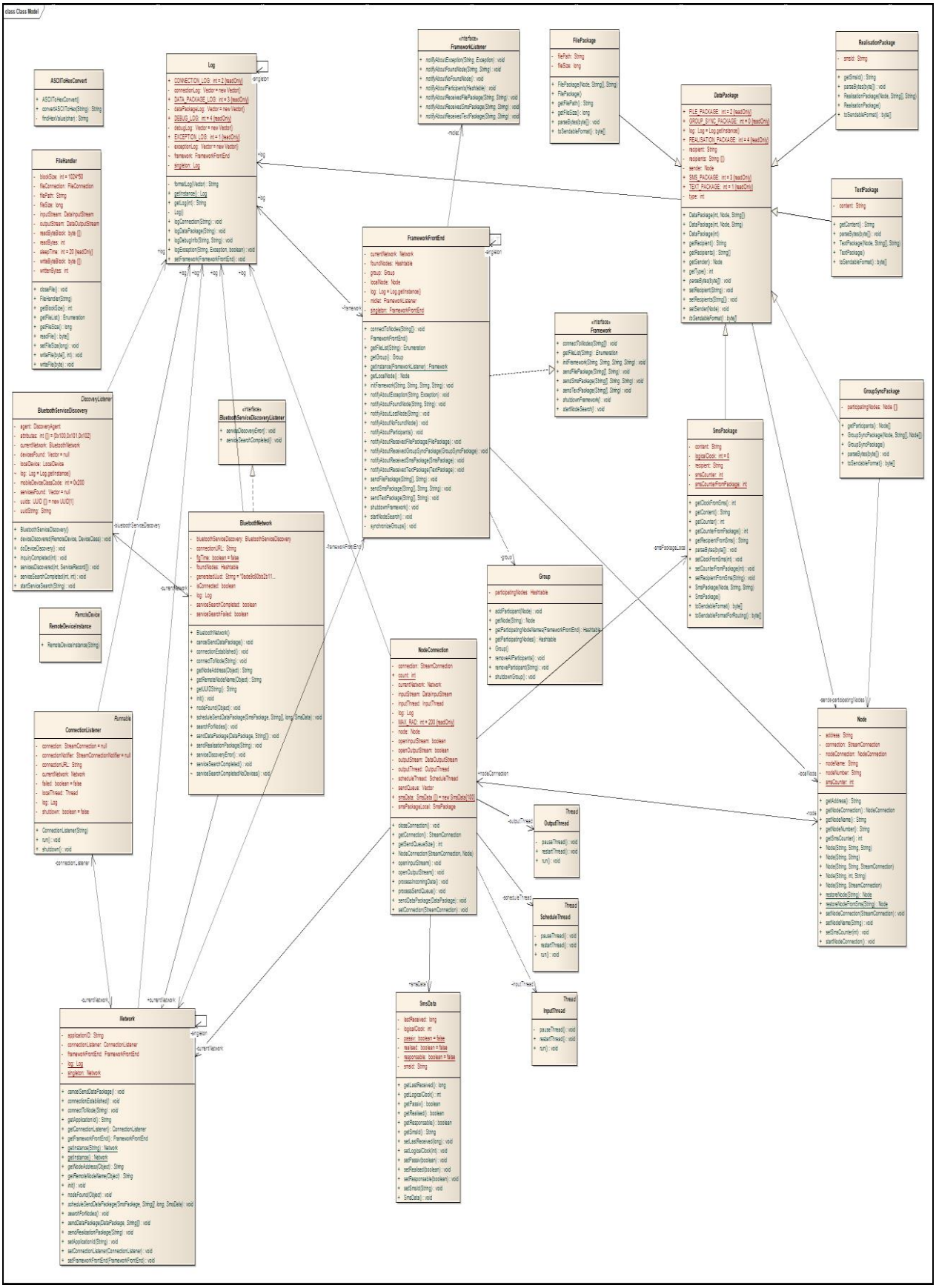

Figura 5.8 Diagrama de clase generalizată

5.3 Pattern-uri utilizate

Această secţiune cuprinde o descriere a design pattern-urilor folosite de această arhitectură. Un pattern reprezintă o soluţie generală la o problemă comună de proiectare folosită pentru generarea unui cod de calitate. Este asemenea unui model de rezolvare a unei probleme care apare în diferite situaţii şi arhitecturi. În mod obişnuit pattern-ul sublinează relaţiile şi interacţiunea dintre obiecte la un nivel ridicat.

Pattern-ul Singleton – Acest pattern este utilizat în programarea orientată pe obiecte pentru a asigura crearea unei singure instanţe a unei anumite clase. Acest lucru este necesar atunci când un singur obiect trebuie utilizat pentru coordonarea acţiuniilor în întreg sistemul. Pattern-ul poate fi implementat prin crearea unei clase cu o metodă care crează o nouă instanţă a unei clase, în cazul în care aceasta nu există. Pentru a obliga celelalte clase să utilizeze această metodă de instanţiere, constructorul clasei este declarat „private" sau „protected". În cazurile folosirii mai multor fire de execuţie pattern-ul singleton este vulnerabil datorită faptului că instanţierea clasei poate fi apelată simultan de două fire de execuţie. Această problema poate fi rezolvată prin sincronizarea metodei getInstance(), realizând astfel excluderea mutuală.

În cadrul arhitecturii Peer2Me acest pattern a fost utilizat pentru clasele Log, FrameworkFrontEnd şi Network pentru a asigura faptul că se crează o singură instanţă a acestor clase.

Pattern-ul Facade – Acest pattern se bazează pe ideea că o faţadă asigură o interfaţă simplificată pentru o porţiune mare de cod. Acest lucru este necesar pentru a face utilizarea, spre exemplu a unei librării de clase, mai uşoară. Se reduc de asemenea dependenţele de cod dintre cele doua părţi interfaţate, făcând mai uşoară modificarea acestora.

Interfaţa Framework a framework-ului Peer2Me a fost proiectată utilizând acest pattern. Un MIDlet care rulează framework-ul are acces doar la metodele acestei interfeţe, care se comportă precum o faţadă între framework şi MIDlet, ascunzând complexitatea framework-ului.

Pattern-ul Observer – Toate clasele GUI J2Me folosesc CommandListener pentru a intercepta acţiuniile utilizatorului. Aceste acţiuni se declanşează în momentul în care un buton este apăsat. Se detectează butonul apăsat şi se execută o anumită operaţie coresptuzătoare acestuia.

Capitolul 6. Implementarea algoritmului de rutare

6.1 Introducere

Rutarea este un termen folosit în cadrul reţelelor de calculatoare pentru a desemna procesul de alegere a căii prin care un pachet este transmis dinspre sursă spre destinaţie prin noduri intermediare, numite rutere. Procesul de rutare directează de obicei pe baza unor tabele de rutare pe care le gestionează ruterele, tabele care conţin o înregistrare a celor mai bune rute către diferite destinaţii din reţea.

6.1.1 Tipuri de mesaje

În funcţie de numărul de staţii care trebuie să primească un anumit mesaj, putem avea:

- Mesaje *unicast*: sunt destinate unei singure staţii;
- Mesaje *multicast*: sunt destinate tututor staţiilor dintr-un grup de staţii identificate de o adresă de multicast;
- Mesaje *manycast*: sunt destinate unui numar arbitrar (specificat de către utilizator) de statii;
- Mesaje *broadcast*: sunt destinate tuturor staţiilor din reţea;
- Mesaje *anycast*: sunt destinate oricarei staţii dintr-un grup de staţii (şi numai uneia).

6.1.2 Definiţie protocoale de rutare

Protocoalele de rutare stabilesc regulile prin care informaţiile despre reţele sunt schimbate între rutere în scopul obţinerii unei tabele de rutare adecvate tipologiei. Reţelele mici pot gestiona tabele de rutare configurate manual. Reţelele mari în schimb implică tipologi mari care se schimbă constant, făcând utilizarea manuală a tabelelor de rutare foarte dificilă.

Există două tipuri mari de rutare care stau la baza tuturor celorlalte tipuri: *rutarea statică şi rutarea dinamică*. Rutarea statică descrie un sistem care rutează într-o reţea de date

în funcţie de căi fixe. Rutarea dinamică construieşte dinamic tabelele de rutare, bazându-se pe informaţiile purtate de protocoale, permiţând reţelei să acţioneze în mod aproape automat pentru a evita erori sau blocaje în reţea. Datorită proprietăţilor sale, rutarea dinamică domină în momentul actual internetul.

Avantajul rutării dinamice faţă de cea statică sunt scalabilitatea şi adaptabilitatea. O reţea rutată dinamic poate creşte mult mai repede şi este capabilă să se adapteze schimbărilor din topologia reţelei aduse tocmai de această creştere sau de erorile din una sau mai multe componente ale reţelei. Într-o rutare dinamică, ruterele învaţă despre topologia reţelei comunicând cu alte rutere. Rutarea dinamică are însă şi dezavantaje, cum ar fi creşterea complexităţii.

Protocoalele de rutare pot fi clasificate şi în funcţie de tipul reţelei, existând protocoale de rutare pentru reţele cablate şi protocoale de rutare pentru reţele wireless. Acest lucru se datorează faptului ca reţelele ad-hoc au pus noi probleme protocoalelor de rutare deja cunoscute, dedicate reţelelor cablate.

Într-o reţea ad-hoc, nodurile reţelei nu au cunoştinţe apriori despre tipologia reţelei din care fac parte astfel încât ele trebuie să o descopere. Ideea de bază este accea că un nou nod (în mod opţional) îşi anunţă prezenţa şi aşteaptă mesaje broadcast de la vecinii săi. Nodul învaţă despre nodurile din apropiere şi despre modul în care poate ajunge la aceste noduri. Odată cu trecerea timpului fiecare nod va şti toate nodurile şi unul sau mai multe moduri de a ajunge la ele.

6.1.3 Clasificarea protocoalelor ad-hoc

a. Rutare pro-activă (Bazată pe tabele de rutare)

Acest tip de protocoale menţin o listă actualizată cu destinaţiile şi rutele lor, prin trimiterea periodică a tabelor de rutare în reţea. Cele mai mari dezavantaje ale unor astfel de algoritmi sunt faptul ca necesită multe resurse pentru menţinerea datelor iar reacţia la schimbările din reţea este foarte lentă. Exemple de astfel de algoritmi sunt AWDS (Ad-hoc Wireless Distribution service), DSDV (Highly Dynamic Destination-Sequenced Distance Vector routing protocol), HSR (Hierarchical State) , Babel inspirat din DSDV ş.a.

b. Rutare reactivă (La cerere)

Acest tip de protocol găseşte o rută la cerere prin inundarea (flooding) reţelei cu un pachet de Route Request. Cele mai importante dezavantaje ale unor astfel de algoritmi sunt

intrevalul mare de timp necesar pentru găsirea rutei, iar inundarea excesivă poate duce la aşa numitul „network clogging”. Exemple de algoritmi „La cerere” sunt Ad-hoc On-demand Distance Vector, Multirate Ad-hoc On-demand Distance Vector Routing Protocol ş.a.

c. Rutare bazată pe flux

Acest tip de protocol găseşte o rută la cerere prin urmărirea fluxului. Una dintre opţiuni este de a trimite mesaje unicast în mod consectiv atunci când se înaintează date în procesul de promovare a unei noi legături. Unul dintre cele mai importante dezavantaje ale unui astfel de algoritm este intervalul mare de timp necesar pentru găsirea unei noi rute fără să existe cunoştiinţe apriori despre tipologia reţelei. Exemple de astfel de algoritmi sunt GB (Gafni-Bertsekas), MOR (Multipath On-demand Routing Protocol), LQSR (Link Quality Source Routing).

d. Rutare dinamică sau adaptativă (Starea legăturii)

Acest tip de protocol combină avantajele rutării proactive şi reactive. Un exemplu de astfel de algoritm este TORA (Temporally-Ordered Routing Algoritm).

e. Rutare ierarhică

În acest tip de protocol alegerea rutării reactive şi proactive depinde de nivelul ierarhic în care se află nodul. Exemple de astfel de protocoale sunt CBRP (Cluster Based Routing Protocol), CEDAR (Core Extraction Distributed Ad-hoc Routing), DART (Dynamic Address Routing).

f. Rutare geografică

Numită şi rutare bazată pe poziţie reprezintă un principiu de rutare care se bazează pe informaţia oferită de poziţia geografică, în ideea că sursa trimite un mesaj adresei geografice a destinaţie în loc să folosească adresa de reţea. Rutarea geografică presupune ca fiecare nod să-şi poată determina propria locaţie şi ca sursa să cunoască locaţia destinaţiei. Exemple de astfel de protocoale sunt ALARM (Adaptative Location Aided Routing Protocol), DREAM (Distance Routing Effect Algorithm for Mobility), GPSAL (DPS Ant-Like Routing Algoritm).

g. Protocoale de rutare care ţin cont de consumul de energie

Energia necesară transmiterii unui semnal este aproximativ proportională cu d^{α} , unde d reprezintă distanta iar $\alpha \geq 2$ reprezintă factorul de atenuare, care depinde de mediul de transmisie. Când $\alpha = 2$ (acesta fiind cazul optim), trasmiterea unui semnal pe jumatate de distanţă necesită o pătrime din energie iar daca există un nod la mijloc care consumă de asemenea o pătrime din energia sa pentru a transmite pachetul pe cealaltă jumatate de distanţă, dateler ar fi transmise cu jumătate din energia necesară unei transmisii directe. Unul dintre dezavantajele majore ale acestui tip de rutare este acelea că această metodă produce o întarziere pentru fiecare transmisie. Exemple de astfel de protocoale sunt ISAIAH (Infra-Structure Aodv for Infrastructured Ad Hoc networks), EADSR (Energy Aware Dynamic Source Routing Protocol), PAMAS (Power Aware Multi Access Protocol whit Signaling Ad Hoc Networks)

h. Rutare multicasting

Exemple de astfel de protocoale de rutare sunt ADMR (Adaptative Demand-Driven Multicast Routing), Scribble, ODMPR (On-Demand Multicast Routing Protocol), SPMB (Scalable Position-Based Multicast). Algoritmul ODMPR este un algoritm de rutare „la cerere” pentru pachete multicast.

i. Rutare geografica multicast

Exemple de astfel de protocoale de rutare sunt LBM (Location Based Multicast), GeoGRID (Geographical Grid), MOBICAST (Mobile Just-in-time Multicasting).

j. Alte clase de protocoale

Exemple de astfel de protocoale de rutare sunt FQMM (Flexible QoS Model for MANET), B.A.T.M.A.N. (Better approach to mobile ad hoc networking).

6.2 Protocolul de rutare Scribble -M

În continuare va fi prezentat protocolul Scribble implementat în cadrul acestei lucrări de licenţă, o modificare a protocol propus şi descris în [12]. Acest protocol este un protocol care garantează faptul că cel puţin un număr k de destinatari vor primi un mesaj trimis într-o reţea wireless. Aceste garanţii sunt deterministe, iar probabilitatea de eşec este zero. Valoarea lui k este aleasă de către aplicaţie astfel încât să reprezinte o valoare realistă. Un astfel de

serviciu numit manycast [13] poate fi folosit într-o colecţie de aplicaţii precum comunicarea între echipele de salvare, publicitate realizabila de proprietarii magazinelor in proximitatea acestora, ş.a. pentru care Scribble ar fi un candidat ideal.

Având ca punct de plecare acest protocol, lucrarea de faţă propune o modificare a acestuia astfel, daca scopul protocolului Scribble este acela de a livra un mesaj la un număr k de destinatari, scopul protocolului propus, implementat si încorporat în framework-ul Peer2Me a fost acela de a livra un *mesaj la un destinatar anume localizat in afara proximitatii specifice tehnologiei Bluetooth a emitatorului.*

Cu toate ca scopurile diferă, ideea de rutare a mesajului în reţea va rămane aceaşi, acesta fiind şi motivul pentru care algoritmul Scribble a fost preferat altor algoritmi de rutare precum ODMPR.

6.2.1 Introducere

Problema trimiterii eficiente a unui mesaj la noduri multiple într-o reţea ad-hoc a fost studiată intens în ultimul timp. Soluţiile propuse includ protocoale de rutare de tip broadcast, multicast şi manycast.

Garanţia faptului că un mesaj a fost trimis tuturor nodurilor din interiorul unui grup presupune asumpţia că nici un nod destinaţie nu părăseşte grupul în timpul execuţiei unui protocol. Fără această asumpţie, încercarea de a trimite un mesaj unui nod care s-a îndepărtat sau a picat, continuă pană în momentul în care se reactualizează reţeaua; aceste încercări ducând la încărcarea reţelei. Nodurile care alcătuiesc o reţea ad-hoc sunt predispuse eşecului datorită resurselor limitate (ex. bateria) sau datorită faptului că se îndepărtează de grup.

Protocolul prezentat, permite unor noduri să se îndepărteze sau să pice în timpul execuţiei. Mai precis, se asigură că k destinatari vor primi mesajul *(k deterministic delivery guarantees*). Valoarea lui k este aleasă de către aplicaţie, astfel încât să fie o valoare realistă. Un astfel de serviciu este numit *manycast.* Orice protocol care asigură garanţii deterministe de livrare a mesajelor trebuie să includă un fel de înştinţare (ack) de primire a mesajului de la destinatarii acestuia. Acest fapt poate conduce la problema “ack-implosion” care este şi mai pronunţată în cazul unei reţele dinamice precum o reţea ad-hoc. Problema a fost rezolvată prin descentralizarea responsabilităţii înştinţării de primire a mesajului de către destinatari.

6.2.2 Definirea problemei

Algoritmul defineşte un grup G format din n noduri N_0, N_1,.., N_{n-1}. Nodurile au capacităţi de procesare, stocare şi comunicare similare, comunicând folosind o transmisie wireless omni-direcţională. Protocolul rezolvă problema unui manycast deterministic sigur, definită astfel:

Considerând faptul că orice nod din G, de exemplu N_0, poate iniţia la un timp t_0 trimiterea unui pachet manycast în reţea, notat m, un protocol manycast satisface următoarele trei proprietăţii:

i. Integritatea: un nod care livrează m (unei aplicaţii de nivel mai înalt), îl livrează doar o dată.
ii. Terminarea: cel puţin k, $1 \leq k \leq n$, noduri din G primesc pachetul m într-un interval de timp limitat de la momentul de timp t_0, unde k este parametrul manycast.
iii. Căderea reţelei: După scurgerea unui anumit interval de timp de la momentul de timp t_0, tramsmiterea lui m sau a oricaror altor pachete legate de m se opreşte.

Prin aceste definiţii, o singură trimitere a unui pachet m ar trebui să ajungă la k destinatari, într-un anumit interval de timp limitat şi cu o limită a cheltuielilor transmisiei.

Cu scopul de a rezolva problema unui manycast deterministic sigur, urmatoarele condiţii trebuiesc îndeplinite:

i. Fiecare nod trebuie sa aibă un ID unic.
ii. Cel puţin k noduri din G trebuie să fie operative pe durata transmisiei manycast.

Prima condiţie este necesară pentru a determina dacă cel puţin k noduri distincte au primit pachetul m. Cea dea doua condiţie este necesară pentru ca protocolul să ajungă la condiţia de terminare, adică ca pachetul sa poată fi livrat la k noduri.

În reţelele ad-hoc, datorită mobilităţii nodurilor, interferenţelor, diferitelor obstacole sau măsurilor de economisire a bateriei este posibil ca un subset de noduri din grup să fie partiţionate de restul nodurilor din grup la orice moment în timp, ceea ce înseamnă că nici unul dintre nodurile subsetului nu vor fi capabile să comunice cu celelalte noduri ale reţelei. Dacă partiţiile sunt permanente, în mod clar condiţia de terminare nu poate fi îndeplinită: dacă N_0 (acesta fiind nodul care a iniţiat transmisia lui m) şi alte cateva noduri sunt într-un subset,

care a fost partiţionat permanent de orice alt subset, atunci nici un nod din celelalte subseturi nu va fi capabil să primească pachetul m. De accea, condiţia mimimă pentru rezolvarea problemei unui manycast deterministic sigur este aceea ca cel puţin un subgrup care conţine minim k noduri operative din G, să nu fie partiţionat permanent. Acestă condiţie poartă numele de „Cerinţa minimă de viaţă" iar protocolul manycast ar trebui să asigure garanţiile necesare atat timp cât reţeaua satisfface această cerinţă.

„Cerinţa minimă de viaţă" - Pentru a defini cerinţa de viaţă minimă în mod cât mai precis, se va defini ce reprezintă o partiţionare permanentă (sau absenţa acesteia) în contextul unei reţele ad-hoc, fie aceea reţea mobilă sau statică. Nodurile pot fi unul în raza celuilalt din două motive posibile: fie pentru faptul că îşi setează puterea de transmisie şi senzitivitatea recepţiei la un nivel destul de mare, fie că nodurile ajung fizic unul în apropierea celuilalt, precum în reţelele MANET.

Fie δ întârzierea maximă asociată cu primirea şi procesarea unui pachet. Dacă două noduri sunt unul în raza wireless a celuilalt fară nici o interferenţă pentru mai mult de 2δ, atunci durata conectivităţii poate fi suficient de lungă pentru ca unul dintre noduri să trimită un pachet de cerere şi să primească un pachet de răspuns de la celalalt nod. Vom spune că două noduri sunt direct conectate unul cu celălalt dacă se află unul în raza wireless a celuilalt fără nici o interferenţă pentru o perioada de timp $\beta+2\delta$, unde $\beta \geq 0$ reprezintă timpul maxim pe care un nod care a primit o cerere alege sa îl ia pană în momentul când trimite răspunsul său.

6.2.3 Cerinţe de proiectare

Orice protocol deterministic trebuie să îndeplinească următoarele trei cerinţe:

i. *Răspândirea mesajului*: Modul în care se obţine acoperirea dorită. Acest protocol se ocupă de această problemă într-un mod distribuit. Responsabilitatea iniţială pentru distribuirea pachetului o are iniţiatorul procesului de manycast şi este pasată pe rând celorlalte noduri, asigurându-se astfel faptul că se realizează acoperirea dorită cu costuri de transmisie cât mai mici. Chiar mai mult, această dirtribuire a responsabilităţii este îndeplinită fără să existe vreun acces la o structură de rutare sau la vreo informaţie legată de topologia reţelei. Acest aspect al protocolului este numit *Distribuirea Responsabilităţii*.

ii. *Deducţia acoperirii*: Presupunând că acoperirea dorită a fost realizată, accest fapt trebuie dedus astfel încât operaţiile de manycast să fie finalizate. Acest protocol asigură faptul că orice nod responsabil deduce finalizarea procesului de manycast odată cu livrarea pachetului la nodul k, şi face acest lucru utilizându-se de informaţia locală într-o manieră distribuită. Acest aspect al protocolului este numit *Realizare*.

iii. *Incercarea de finalizare*: Când condiţiile reţelei sunt extrem de ostile, orice schemă folosită pentru (i) nu poate garanta acoperirea dorită. Cu toate acestea un protocol determinist nu ar trebui să se termine pană când sau decât în momentul în care acoperirea dorită a fost obţinută. Astfel, un algoritm determinist ar trebui să apeleze la o masură mai agresivă şi de aceea mai costisitoare atunci când schema folosită este considerată ineficientă. Măsura la care acest protocol apelează, garantează succesul atât timp cât condiţia de viaţă minimă este garantată. Acest aspect al protocolului poartă numele de *Finalizare datorată Conditiei Minime*.

6.2.3.1 Distribuirea Responsabilităţii

Dacă un nod este responsabil pentru m, îl va transmite pe m la fiecare β secunde. Numărul nodurilor responsabile pentru m trebuie menţinut scăzut, în special în cazurile în care mărirea numărului de noduri responsabile nu conduce la o acoperire mai bună. Schema care are ca scop distribuirea responsabilităţii realizează acest lucru prin încercarea de a menţine cel mult un nod responsabil în fiecare subset de noduri conectate în mod direct.

Schema de distribuire a responsabilităţii se foloseşte de următoarele rezultate cunoscute:

R1: Acoperirea adiţională care se aşteptată să rezulte din transmisia unui nod scade exponenţial cu numărul de transmisii care au avut loc în raza wireless a nodului respectiv cu puţin timp înaintea acelei transmisii.

R2: Se consideră două noduri cu ceasuri ale căror valori nu scad niciodată. Mesajele transmise de aceste noduri vor fi marcate cu valoarea ceasului local. Dacă nodurile primesc unul mesajul celuilalt, nu este posibil ca ambele ceasuri locale ale nodurilor să fie mai mici decât timpul marcat pe mesajele pe care le primesc.

În cadrul acestui protocol, ceasul local al unui nod este reprezentat de un *ceas logic*. Un ceas logic este un număr întreg a cărui valoare poate doar să crească, fără să fie în relaţie

cu trecerea efectivă a timpului. Un nod N_i construieşte un ceas logic $L_i(m)$ cu valoarea iniţială zero când N_i i-a pentru prima dată cunoştinţă de m. Dacă N_i este responsabil pentru m, încrementează $L_i(m)$ cu 1 şi îl marchează pe m cu valoarea lui $L_i(m)$ chiar înainte de a transmite mesajul. Acesta reprezintă marca de timp logică (notată cu m.l) a mesajului m transmis.

Presupunem faptul că un nod N_j care nu este responsabil pentru m îl primeşte pe m. N_j va deveni responsabil pentru m dacă:

A1: Valoarea m.l a pachetului m primit este mai mare sau egală cu $L_j(m)$, şi

A2: N_j nu l-a mai primit pe m în ultimele β-δ secunde.

Odată devenit responsabil, N_j setează valoarea lui $L_j(m)$= m.l a pachetului primit, alege o valoare RAD (Random Assessment Delay) uniform distribuită pe intervalul (0, MAX_RAD), şi programează prima dintre transmisile sale periodice ale lui m (incluzând încrementarea lui $L_j(m)$) care să aibă loc la expirarea RAD-ului ales. Dacă N_j a primit o alta transmisie a lui m în ultimele β-δ secunde înseamnă că cel puţin două noduri din raza wireless a lui N_j au devenit responsabile pentru m în trecutul apropiat. În conformitate cu R1, N_j nu devine responsabil în acest caz.

Dacă un nod N_i îl primeşte pe m astfel încât (A1) m.l a pachetului primit $m < L_i(m)$, atunci nodul renunţă la responsabilitatea pentru m şi anulează transmisia programată pentru m. În cazul în care N_i urmează să transmită pentru prima dată mesajul m la expirarea unui RAD, această regulă poate provoca ca nodul N_i să nu transmită niciodată pachetul m.

Când două noduri responsabile îşi primesc unul celuilalt pachetul m, datorită lui R2, doar unul dintre cele doau noduri va renunţa la responsabilitate, cu condiţia ca valorile ceasurilor cu care este marcat pachetul m să nu fie identice. Acest lucru este însă puţin probabil din două motive: odată pentru că nodurile responsabile au în general aceeaşi valoare a ceasului dacă devin responsabile primind copii ale aceluiaşi m, setând astfel valoarea lui L(m) cu valoarea m.l a pachetului primit, iar în al doilea rând datorită faptului că intervalul de timp pentru realizarea primei transmisii este ales aleator. Cu o valoare mare pentru MAX_RAD, este puţin probabil ca două noduri să transmită în acelaşi moment, şi deci să transmită pachetul m cu aceelaşi m.l.

Când un nod îl are pe m dar nu este responsabil pentru el, se spune că nodul respectiv este passiv în ceea cel priveşte pe m.

6.2.3.2 Realizarea trimiterii pachetului

Protocolul cere ca fiecare nod N_i trebuie să menţină o listă, $K_i(m)$, care să conţină o semnatură unică a nodurilor despre care N_i ştie că l-au primit pe m. Când N_i îl transmite pe m, adaugă $K_i(m)$ unui câmp dedicat m.k din antetul pachetului. Dacă N_i primeşte un pachet m a cărui câmp m.k nu conţine semnătura sa, şi se decide să nu transmită pe m, va transmite un mic pachet de înştinţare pentru m. Aceste pachete nu vor provoca o explozie de înştinţări (ack-implosion), deoarece pachetele vor călători un singur hop.

Următoarele reguli îi vor permite lui N_i să finalizeze trimiterea lui m:

1. N_i consideră că m şi-a atins scopul dacă $K_i(m)$ conţine k semnături diferite de semnătura proprie;
2. În timpul finalizării, N_i trimite un pachet de Realizare pentru m care conţine ID-ul lui m;
3. N_i finalizează trimiterea lui m atunci când primeşte un pachet de Realizare pentru m. Primirea unui pachet de realizare însă nu îl va determina pe N_i să trimită un pachet de realizare propriu.

6.2.3.3 Finalizarea datorită Condiţiei Minime

În multe situaţii, o reţea ad-hoc poate necesita mai mult decât împărţirea responsabilităţii pentru asigurarea acoperiri necesare.

Considerând un subset C având toate nodurile direct conectate între ele, fie c_i un nod care iniţiază o transmitere manycast a lui m. Din moment ce $|C| \leq k$, mai multe noduri precum $d \notin C$, trebuie să-l primească pe m. În absenţa unui astfel de d care să intre în raza wireless a unui nod $c_i \in C$, este uşor de observat faptul că nu se va putea obţine acoperirea dorită.

Exprimându-ne altfel, acoperirea dorită se va obţine doar în cazul în care un nod $c_i \in C$ are în raza sa wireless unul sau mai multe noduri precum $d \notin C$, pentru o perioadă de cel puţin $\beta+2\delta$ secunde şi dacă pe parcursul acestei perioade de conectivitate directă nodul c_i este responsabil. Prima condiţie trebuie să fie îndeplinită de reţea, pe când cea de-a doua condiţie trebuie să fie asigurată de către protocol. Se poate observa faptul că o reţea ad-hoc care satisface condiţia minimă de viaţă poate alege orice nod $c_i \in C$ şi să conecteze nodul c_i ales cu d în diferite momente de timp. Într-adevăr, reţeaua ad-hoc se poate comporta ca un adversar, conectând cele două noduri atunci când c_i ales se află în starea de pasivitate faţă de m. Asta înseamnă faptul că un protocol care menţine un singur nod într-un subset C responsabil pentru transmiterea mai departe a mesajului, deşi inteligent proiectat, nu poate asigura faptul că nodurile potrivite din C vor fi responsabile la momentele potrivite. Astfel, măsura pe care o i-

a acest protocol este aceea de a permite tuturor nodurilor din subset sa devină responsabile atunci cand protocolul manycast pare să nu se mai termine.

Fiecare nod N_i are un parametru θ_i. Dacă $L_i(m) \geq \theta_i$ sau daca N_i îl primeşte pe m având $m.l \geq \theta_i$, atunci N_i devine responsabil pentru m.

6.2.4 Descrierea protocolului

Fiecare nod menţine un cache, msg_cache, cu toate pachetele de care nodul este momentan responsabil, şi menţine o structură de date LastRecv(m) conţinând timpul la care a fost primită ultima copie a lui m. Predicatele responsable(m), passive(m) şi realized(m) sunt setate ca fiind adevărate dacă nodul este responsbil pentru, sau este passiv în privinţa lui m, sau m a fost realizat. În plus fiecare nod menţine un ceas logic, L(m), pentru fiecare pachet pe care l-a primit, şi menţine înştinţarea, K(m) care conţine nodurile care l-au primit pe m.

Presupunem că fiecare nod este capabil să realizeze transmisii wireless, şi poate programa o astfel de transmisie pe viitor, sau poate anula o transmisie programată în cazul în care aceasta nu a fost realizată încă. Aceste cerinţe ar trebui să poată fi îndeplinite de orice dispozitiv din reţeaua ad-hoc.

Pentru a putea realiza o transmisie multicast a unui pachet, nodul care iniţiază transmisia apelează metoda RBCast() care actualizează structurile de date necesare şi programează o transmisie.

Algoritmul 1 RBCast()

```
RBCast()
{
        m.id = {my_id, seq#};
        K(m) + = my_id;
        L(m) = m.l = 0;
        Msg_cache.add(m);
        ScheduleTransmit(m.id,NOW);
}
```

Transmit() dacă nu este anulată, transmite m şi se reprogramează după β secunde.

Algoritmul 2 Transmit()

```
Transmit(m.id)
{
        m.K = K(m);
        L(m) = m.l = L(m)+1;
        wireless_transmit(m);
        ScheduleTransmit(m.id, β);
}
```

Dacă un nod recepționează o realizare sau o înștințare de pachet, RBReceiveRealization() sau RBReceiveAck() va fi apelat.

Algoritmul 3 RBReceiveAck() și RBReceiveRealization()

```
RBReceiveAck(Acknowledgment ack)
{
        K(m) += ack.id;
        if(realized(ack.id))
        {
                transmitRealizationPacket(ack.id));
                cancel pending transmits(ack.id)
                delete K(ack.id);
                msg cache.delete(ack.id);
                realized(ack.id) = TRUE
        }
}

RBReceiveRelization(RealizationPacket real)
{
        msg cache.delete(real.id);
        cancel pending transmits(real.id);
        delete K(real.id);
        realized(real.id) = TRUE;
}
```

În final, când pachetul de date a fost primit, RBReceive() este apelat.

Algoritmul 4 RBReceive()

```
RBReceive(DataPacket m)
{
        if(m.K !contain my id)
        {
                transmitAck(m.id);
        }
        if(L(m) undefined)
        {
                L(m) = m.l;
                K(m) = m.K;
```

```
            K(m)+= my id;
            deliver(m);
        }
        K(m) = m.K + K(m);
        if(realized(m.id))
        {
            transmitRealizationPacket(m.id));
            delete K(m.id);
            msg cache.delete(m.id);
        }
        else if(passive(m.id))
        {
            if((LastRecv(m.id) + (!-'') < NOW     && m.l >=L(m)))
            {
                L(m) = m.l;
                msg cache.add(m);
                ScheduleTransmit(m.id,
                random(0, MAX RAD));
                responsible(m.id) = TRUE;
            }
        }
        else if(responsible(m.id))
        {
            if(m.l > L(m) && m.l < #)
            {
                L(m) = m.l;
                msg cache.delete(m.id);
                cancel pending transmits(m.id);
                passive(m.id) = TRUE;
            }
        }
        LastRecv(m.id) = NOW;
}
```

6.2.5 Implementarea protocolului

6.2.5.1 De ce Scribble? Argument

Motivul alegerii protocolul de rutare Scribble a fost oferit de faptul că acest algoritm nu necesită menţinerea unei structuri de date pentru rutare, precum tabelele de rutare. Două dintre caracteristicele de baza ale reţelelor ad-hoc formate din telefoane mobile sunt puterea scăzută de procesare şi capacitatea mică de stocare, de unde rezultă şi faptul că utilizarea unor protocoale de rutare bazate pe menţinerea anumitor structuri necesare rutării pachetelor nu

sunt adecvate acestor dispozitive. De asemenea, utilizarea unor structuri de date pare inutilă datorită mobilităţii sporite a acestor dispozitive.

Un al doilea motiv pentru care am ales acest protocol a fost studiul comparativ [12] realizat între acest protocol atunci când k = n (broadcast) şi protocolul ODMPR, demonstrând faptul că mecanismul de rutare propus de acest protocol nu presupune neaparat cheltuieli suplimentare la nivelul infrastructurii de rețea.

6.2.5.2 Modificări aduse protocolului:

Deşi protocolul Scribble descrie o rutare de tip de manycast, versiunea implementată în framework-ul Peer2Me va efectua o rutare de tip unicast.

Condiţia de finalizare a protocolului, constând în livrarea unui pachet la k destinatari, a fost modificată astfel încât protocolul să realizeze o operaţiune unicast, condiţia de finalizare constând în livrarea pachetului la un anumit destinatar.

Datorită faptului că scopul protocolului a fost modificat, condiţia de finalizare fiind îndeplinită în momentul în care pachetul ajunge la o anumită destinaţie, lista K(m) a fost considerată inutilă în noul context, renunţându-se în consecinţă la acestă listă. De asemenea, trimiterea pachetelor de înştinţare de primire a mesajului îşi pierd utilitatea.

Necesitatea algoritmului ca fiecare nod să fie identificat printr-un Id unic se justifică în noul context prin nevoia de a identifica nodul destinaţie al pachetului. În cadrul framework-ului Id-ul unic va fi reprezentat de numărul de telefon al dispozitivului, acesta fiind un număr unic pentru fiecare telefon mobil. Datorită faptului că un telefon mobil nu îşi cunoaşte propriul număr de telefon, responsabilitatea asignării numărului de telefon dispozitivului va fi transferată utilizatorului acelui dispozitiv.

6.2.5.3 Modificări aduse framework-ului

Modificările efectuate asupra framework-ului constau în crearea de noi clase care să asigure noua funcţionalitate a framework-ului, dar şi în modificarea vechilor interfeţe şi clase pentru a putea implementa corespunzător algoritmul de rutare.

Pentru a putea prezenta într-un mod cât mai structurat modificările făcute voi prezenta mai întâi paşii pe care un pachet trebuie să-i urmeze din momentul crearii pachetului şi până în momentul finalizării algoritmului, determinand livrarea pachetului la nodul destinaţie:

Pasul 1. Crearea condiţiilor necesare transmiterii unui mesaj unicast:

Pentru ca un nod să poată iniţia transmiterea unui mesaj utilizând protocolul implementat, este necesar ca acest nod să fie identificabil printr-un Id unic reprezentat de numarul de telefon al dispozitivului. Pentru ca acest lucru sa fie posibil, am introdus un nou atribut în clasa *Node* din pachetul *Domain*, atribut numit *nodeNumber* care să conţină numarul de telefon introdus de către utilizator prin intermediul MIDlet-ului instalat pe telefonul mobil.

O a doua condiţie înainte de a începe creearea unui pachet unicast, o reprezintă necesitatea ca nodul să facă parte dintr-un grup, adică să fie direct conectat cu alte noduri din raza sa wireless.

O dată îndeplinite aceste două condiţii devine posibilă crearea unui mesaj unicast.

Pasul 2. Creearea unui nou mesaj:

Pentru a realiza funcţionalitatea trimiterii unui mesaj la un destinatar din afara grupului, implementând astfel ideea de scatternet, am definit mai întâi în interfaţa *Framework* o nouă metodă denumită *sendSmsPackage*. Acest lucru se datorează faptului că metodele definite de această interfaţă sunt singurele metode vizibile midletului. Clasa care implementează aceste metode este clasa *FrameworkFrontEnd*. Împlementarea metodei *sendSmsPackage* realizează crearea unui nou pachet unicast folosind informaţiile primite de la MIDlet în acest scop, şi anume numărul de telefon al destinatarului şi conţinutul mesajului, şi totodată trimite noul pachet în reţea.

Pentru a creea un pachet unicast am creat o nouă clasă în interiorul pachetului *Domain*, numită *SmsPackage*, clasa care moşteneşte clasa *DataPackage* la fel ca şi clasele *TextPackage, FilePackage* şi *GroupSyncPackage*. Metoda clasei *toSendableFormat* creează un pachet de forma:

from:”numele-expeditorului”:”numărul-mesajului”:”numărul-de-telefon-al-expeditorului”
to:”numărul-de-telefon-al-detinatarului”
content:”conţinutul”
clock:"ceasul-logic-al-mesajului"

pe care îl transformă apoi într-un şir de biţi care poate fi trimis prin reţea.

Numărul mesajului este de fapt un contor al mesajelor iniţiate de către nodul expeditor, şi a fost introdus pentru a obţine un Id unic al mesajului, care este de forma: „*număr-telefon-expeditor”*: „*număr-telefon-destinatar*”: „*număr-mesaj-extras-din-mesajul_unicast*”. Motivul pentru care am considerat importantă introducerea acestui contor de mesaje este acela ca protocolul să nu confunde mesajele trimise de aceeaşi expeditor la acelaşi destinatar la intervale mici de timp.

Ceasul logic al mesajului reprezintă însuşi ceasul logic necesar protocolului de rutare. Acesta va fi incrementat de fiecare dată cand un mesaj este planificat pentru a fi trimis în reţea în urma protocolului de rutare, chiar înainte de expirarea intervalului de timp β.

Pasul 3. Trimiterea mesajului în reţea:

Acest pas este realizat de clasa *BluetoothNetwork*, care moşteneşte clasa *Network*. Aşa cum reiese din afirmaţiile anterioare, trimiterea pachetului în reţea se realizează prin apelul metodei *sendDataPackage*, implementată de clasa *BluetoothNetwork*, din cadrul metodei *sendSmsPackage* din clasa *FrameworkFrontEnd*. Este important de precizat faptul ca pachetul va fi transmis tuturor nodurilor din reţea.

În metoda *sendDataPackage* se verifică conexiunea la nodurile destinaţie, stabilindu-se o legatură cu acestea în cazul în care aceasta nu există, iar apoi se apelează metoda *sendDataPackage* din clasa *NodeConnection*. Aceasta adaugă mesajul la coada de tip FIFO de ieşire (*sendQueue*) iar apoi are loc înştinţarea firului de execuţie care „ascultă” într-un ciclu neinfinit coada de iesire. Firul de execuţie va apela apoi metoda *processSendQueue* din interiorul aceleaşi clase, trimiţând într-un final pachetul în reţea.

Echivalentul la msg_cache din cadrul algoritmului de rutare este noua clasa *SmsData*, care ţine toate informaţiile despre un pachet m. Clasa a fost definită în pachetul *Util* iar un obiect de tipul SmsData poate fi identificat în funcţie de Id-ul mesajului. Un alt atribut al clasei este ceasul logic corestunzător pachetului (*logicalClock*). De asemenea, sunt declarate atributele logice *realised*, *responsable* şi *passiv* care ţin locul predicatelor realized(m.id), responsible(m.id) şi passive(m.id) definite de algoritmul Scribble. Predicatul LastRecv(m.id) are echivalentul în atributul *lastReceived* aparţinând aceleaşi clase.

Astfel, după trimiterea pachetului în reţea, se crează un nou obiect într-o listă statică de tipul *SmsData* având *logicalClock*-ul setat la 0. Acest lucru este necesar pentru cazul în care pachetul ajunge din nou la nodul expeditor în cadrul procesului de rutare.

După instanţierea clasei *SmsData*, se apelează metoda *scheduleSendDataPackage*, care planifică o nouă transmitere a mesajului după un interval ales aleator din interval (0,MAX_RAD). Această metodă a fost definită în clasa *BluetoothNetwork*.

Pasul 4. Primirea unui mesaj din reţea:

În momentul primirii unui nou mesaj din reţea, firul de execuţie înştinţat de primirea unui nou mesaj apeleaza funcţia *processIncomingData* din interiorul clasei *NodeConnection*. Dacă tipul mesajului primit este SMS_PACKAGE, atunci, după citirea şirului de biţi de intrare, se crează un obiect gol de tipul *SmsPackage*. Dupa crearea obiectul se apeleaza metoda *parseBytes* din clasa *SmsPackage*, metodă care extrage informaţiile din mesajul primit.

Cazul I: Mesajul a ajuns la nodul destinaţie

Dacă numărul de telefon al destinatarului din mesaj este acelaşi cu cel al nodului local atunci pachetul a ajuns la destinatar şi ca urmare MIDlet-ul va fi înştinţat prin intermediul metodei *notifyAboutReceivedSmsPackage* implementată în clasa *FrameworkFrontEnd* şi definită în interfaţa *Framework* în legătură cu noul mesaj primit, iar apoi un *RealisationPackage* va fi trimis la toate nodurile direct conectate la nodul local.

Crearea unui *RealisationPackage* se face prin instanţarea noii clase *RealisationPackage* definită în pachetul *Domain*. Structura unui *RealisationPackage* conţine doar Id-ul mesajului, acest pachet fiind folosit pentru finalizarea algoritmului. Pachetul de finalizare va călători un singur hop.

Cazul II: Mesajul trebuie rutat

În cazul în care pachetul a ajuns la un nod intermediar se verifică dacă pachetul a mai fost primit anterior. Pentru a face această verificare se caută Id-ul mesajului în lista de obiecte SmsData.

Dacă se găseşte un obiect cu Id-ul mesajului, se verifică dacă nodul este responsabil sau passiv în privinţa mesajului.

Altfel se crează un nou obiect SmsData, căruia i se setează atributele cu informaţiile extrase din mesajul primit. De asemenea atributul *responsable* va fi setat True, astfel încât nodul va deveni responsabil pentru pachetul primit, şi ca urmare va planifica transmiterea pachetului după un interval de timp β, ales aleator din intervalul (0,MAX_RAD).

Pentru ca algoritmul să funcţioneze într-un mod corect a fost declarat un nou fir de executie de tipul *ScheduleThread* având prioritate normala. Spre deosebire de firele de execuţie *OutputThread* şi *InputThread* care au prioritate mare, acest fir de execuţie a fost declarat cu prioritate normală datorită faptului că primirea unui nou mesaj, spre exemplu un *RealisationPackege*, este mai importantă decât planificarea de trimitere a unui alt mesaj. Clasa *ScheduleThread* este o clasă abstractă declarată în clasa *NodeConnection*. Firul de execuţie nou creat este pornit sau repornit în cazul unui pachet de tipul *smsDataPackage* care trebuie planificat pentru a fi trimis în reţea dupa intervalul de timp β.

Pasul 5. Primirea unui mesaj de finalizare:

În momentul în care mesajul a ajuns la nodul destinaţie se trimite un mesaj de finalizare conţinând Id-ul mesajului primit. Acest mesaj de finalizare călătoreşte un singur hop în reţea. În cazul în care un nod a primit un astfel de mesaj, nodul respectiv setează atributul *passiv* pe valoarea True şi apeleaza metoda *cancelSendDataPackage* implementată de clasa *BlutoothNetwork*. Apelul acestei metode va rezulta în finalizarea procesului de trimitere a mesajului folosind algoritmul de rutare Scribble.

Partea IV

Aplicaţia PeerMessenger

Capitolul 7. Dezvoltarea unui MIDlet

O aplicaţie Java dezvoltată pentru un dispozitiv CLDC poartă numele de MIDlet. Acest MIDlet trebuie transformat într-un fisier JAR (Java Archive) pentru a putea rula pe un dispozitiv mobil. Fişierele JAR reprezintă versiunea Java a fişierelor ZIP şi pot fi deschise folosind WinZip sau WinRar. În mod normal, funcţia de creare a fişierului JAR este asigurată de unealta de dezvoltare, dar poate fi realizată şi manual folosind executabilul java.exe conţinut in Java2 Standard Edition Software Development Kit (J2SE SDK). Fiecare JAR trebuie să aibă un fişier metadata asociat. Aceste fişiere metadata sunt numite Java Application Descriptor (JAD) şi conţin informaţii pe care Java Application Manager (JAM) le utilizează pentru verificarea şi configurarea MIDlet-ului în momentul rulării [14].

Pentru a rula un MID-let pe un dispozitiv mobil, fişierul JAR care conţine MIDlet-ul compilat trebuie transferat pe dispozitiv. Acest transfer poate fi realizat utilizând un cablu de date între calculator şi dispozitiv, sau prin Bluetooth sau IrDA. Utilizarea Blutooth sau IrDA presupune ca ambele dispozitive să dispună de aceste tehnologii. După realizarea transferului, fişierul JAR poate fi executat, realizându-se astfel instalarea MIDlet-ului pe dispozitiv.

7.1 Cerinţele necesare realizării unui MIDlet

Pentru a face posibilă realizarea unui MIDlet utilizând framework-ul Peer2Me sistemul care dezvoltă MIDlet-ul trebuie să dispună de următoarele tehnologii:

Jar-ul care conţine framework-ul Peer2Me: Acest fişier trebuie să se gasească în calea utilizată pentru compilarea codului corespunzător MIDlet-ului.

Java2 Standard Edition Software Development Kit (J2SE SDK): Acest sdk suportă dezvoltarea de aplicaţii J2SE. Motivul pentru care este necesar acest kit este acela de a avea acces la fişierul *jar.exe*. Executabilul este folosit pentru a putea citi conţinutul jar-ului framework-ului Peer2Me în momentul compilării MIDlet-ului.

Java2 Micro edition (J2ME): J2Me este o ediţie Java special proiectată pentru telefoane mobile, PDA-uri şi alte dispozitive mobile. J2ME asigură un subset de clase şi metode disponibile în Java2 Standard Edition (J2SE).

7.2 Etapele necesare realizării unui MIDlet

Această secţiune descrie paşi necesari realizării unui MIDlet care utilizează framework-ul Peer2Me.

7.2.1 Iniţializarea Framework-ului

Pentru a putea accesa funcţionalităţiile framework-ului, o instanţă a framework-ului trebuie creată. Obţinerea aceastei instanţe şi iniţializarea framework-ul se va realiza parcurgând următoarele etape:

1. Clasa principală a MIDlet-ului trebuie să extindă clasa *javax.microedition.midlet.MIDlet*.
2. MIDlet-ul trebuie să importe *peer2me.framework.**.
3. De asemenea, MIDlet-ul trebuie să implementeze metodele interfeţei *peer2me.framework.FrameworkListener*. Acestă interfaţă este utilizată pentru a trimite date înapoi la MIDlet.
4. Obţinerea unei referinţe de tipul *Framework* se realizează prin apelul metodei *getInstance* din clasa FrameworkFrontEnd. Se recomandă ca acest apel să fie realizat în constructorul clasei principale a MIDlet-ului, iar referinţa obţinută să fie salvată într-o variabilă globală.
5. După obţinerea referinţelor necesare se va apela metoda *initFramework*. Acestă metoda iniţializează framework-ul şi provoacă crearea şi pornirea tuturor firelor de execuţie care se ocupă de diferite funcţionalităţi ale reţelei.

7.2.2 Setarea unei Conexiuni

Pentru căutarea altor dispozitive şi pentru setarea conexiunii cu aceste dispozitive trebuie realizate următoarele etape:

1. Se rulează metoda *startNodeSearch* folosind referinţa framework-ului. Această metodă va începe căutarea unor dispozitive care rulează acelaşi MIDlet.
2. Dacă un astfel de dispozitiv a fost găsit, este apelată metoda *notifyAboutFoundNode* declarată de interfaţa *FrameworkListener*, pentru a anunţa MIDlet-ul cu privire la nodul găsit. Această metodă implementată

de către MIDlet va afişa numele nodului găsit într-un grup care conţine nodurile participante la reţea.

3. După selecţionarea nodurilor la care se doreşte conectarea din grupul cu noduri disponibile, este necesară realizarea unui apel al metodei *connectToNodes* având ca parametru de intrare adresele nodurilor alese. Metoda conectează dispozitivele realizând o reţea ad-hoc.
4. Odată ce conexiunile au fost stabilite, metoda *notifyAboutParticipants* specificată de interfaţa *FrameworkListener* este apelată pe toate dispozitivele conectate, pentru a înştinţa MIDlet-urile cu privire la nodurile participante la reţea. Numele participanţilor vor fi adăugate la grupul cu noduri disponibile de pe fiecare dispozitiv.

7.2.3 Trimiterea unui Pachet de date

Trimiterea unui pachet de date în reţea a devenit un proces foarte simplu de realizat:

1. Pentru trimiterea unui simplu text la nodurile din proximitatea nodului expeditor trebuie folosită metoda *sendTextPackage*. Acestă metodă trimite respectivul text în reţea. Metoda necesită ca parametri de intrare o listă cu adresa nodurilor destinatare şi textul care trebuie trimis.
2. În momentul în care pachetul ajunge la nodurile destinatare, acestea sunt înştinţate de metoda *notifyAboutReceivedTextPackage* specificată de interfaţa *FrameworkListener*. MIDlet-ul local fiecărui destinatar trebuie în consecinţă să afişeze textul primit.
3. Pentru trimiterea unui fişier la nodurile din proximitatea nodului expeditor trebuie folosită metoda *sendFilePackage*. Acestă metodă trimite respectivul fişier în reţea. Metoda necesită ca parametri de intrare o listă cu adresa nodurilor destinatare şi calea spre fişierul care trebuie trimis.
4. În momentul în care pachetul ajunge la nodurile destinatare, acestea sunt înştinţate de metoda *notifyAboutReceivedFilePackage* specificată de interfaţa *FrameworkListener*. MIDlet-ul local fiecărui destinatar trebuie în consecinţă să afişeze un text care să anunţe primirea unui nou fişier.
5. Pentru trimiterea unui sms la un nod anume, identificat prin numărul de telefon, trebuie folosită metoda *sendSmsPackage*. Acestă metodă trimite respectivul sms la nodul destinatar utilizând algoritmul de rutare implementat în cadrul acestei lucrări de diplomă. Metoda necesită ca

parametri de intrare o listă cu adresa nodurilor din proximitatea expeditorului, numărul de telefon al destinatarului şi textul mesajului care trebuie trimis.

6. În momentul în care pachetul ajunge la nodul destinatar, acesta este înştinţat prin metoda *notifyAboutReceivedSmsPackage* specificată de interfaţa *FrameworkListener*. MIDlet-ul local al destinatarului trebuie în consecinţă să afişeze textul mesajului primit.

7.2.4 Utilizarea Log-ului

Framework-ul are un log propriu, foarte util în timpul dezvoltării şi testării MIDlet-ului. Log-ul este utilizat în tot framework-ul, fiind umplut cu diferite tipri de mesaje care informează utilizatorul (şi/sau dezvoltatorul) cu privire la evenimentele care au avut loc în timpul rulării. Pentru utilizarea log-ului trebuie realizate următoarele etape:

1. Mai întâi trebuie importată clasa *peer2me.util.Log*.
2. Referinţa la clasa Log se obţine prin apelul metodei statice *getLog*. Afişarea conţinutului instanţei acestei clase poate fi foarte folositoare, mai ales în cazul apariţiei unor excepţii.
3. Noile mesaje adaugate la log pot fi de asemenea afişate de către MIDlet.

Capitolul 8. Aplicaţia PeerMessenger

Aplicaţia PeerMessenger are la bază unul din exemplele oferite în specificaţiile versiunii 2.0 ale framework-ului. Exemplu în cauză realizează o aplicaţie gen „Chat” pentru utilizatorii acestui servici aflaţi în aceeaşi proximitate şi totodată făcând parte din acelaşi grup.

Noua aplicaţie păstrează vechea funcţionalitate a MIDlet-ului, căruia îi adaugă posibilitatea trimiterii de mesaje gen „sms” la un destinatar specificat printr-un număr de telefon. Pentru realizarea acestei funcţionalităţii au fost dezvoltate noi interfeţe, noi comenzii, printre care şi o comandă de „Refresh”, care actualizează lista participanţiilor la grup. Aceasta opţiune a fost introdusă ca rezultat al testărilor efectuate, datorită faptului că exista o inconsistenţă a grupului pe telefoanele utilizate în procesul de testare.

Codul corespunzător aplicaţiei poate fi găsit pe CD-ul asociat acestei lucrări de diplomă.

Capitolul 9. Testarea aplicaţiei

Procesul de testare a aplicaţiei s-a desfaşurat pe tot parcursul implementării algoritmului de rutare folosind emulatorul Sun Java Wireless Toolkit 2.5.2 pentru CLDC. Acest emulator este cuprins în IDE-ul NetBeans versiunea 6.0.1 utilizat în procesul de optimizare a framework-ului şi include o serie de capabilităţi, precum suport pentru MIDP 2.1, CLDC 1.1, Bluetooth (JSR-82), Web Services (JSR-172), 3D grafic (JSR-184). Versiunile utilizate pentru optimizarea framework-ului au rămas însă MIDP 2.0 şi CLDC 1.1 datorită faptului ca există puţine telefoane care dispun de versiunea 2.1 pentru MIDP. Simularea aplicaţilor pe acest emulator reprezintă un suport important în optimizarea framework-ului şi/sau dezvoltarea de noi aplicaţii.

Testarea practică a aplicaţilor poate ridica însă probleme noi, care nu au existat în timpul simulării pe emulator, probleme datorate diferenţelor de resurse dintre calcularoare şi dispozitivele mobile, precum capacitatea memoriei, puterea de procesare, interferenţele ş.a. Datorită resurselor limitate, una dintre cele mai diferenţe este legată de performanţă. De asemenea, rularea unui număr de până la patru fire de execuţie este suportată de majoritatea telefoanelor mobile, în timp ce rularea unui număr de aproximati zece fire de execuţie provoacă blocarea telefoanelor mobile.

Una dintre cele mai mari probleme apărute în procesul de testare a aplicaţiilor J2ME s-a datorat implementării neuniforme a J2ME pe dispozitivele mobile prezente pe piaţă. Testarea efectivă a aplicaţiilor s-a realizat utilizând tefefoane marca Nokia, acestea îndeplinind condiţiile necesare rulării optime, precum MIDP 2.0, CLDC 1.1, Bluetooth, resurse necesare de memorie dar şi putere suficientă de procesare.

Partea V

Evaluare

Capitolul 10. Evaluarea Framework-ului Peer2Me

Acest capitol prezintă câteva aspecte legate de evaluarea framework-ului Peer2Me. Scopul principal al framework-ului, specificat şi în capitolele introductive ale acestei lucrări de licenţă, este acela de a oferi suport în dezvoltarea aplicaţiilor colaborative pentru telefoane mobile, bazate pe J2ME şi orice protocol de reţea disponibil. Limitarea la tehnologia Bluetooth se datorează faptului că este singura tehnologie de reţea care deţine un API implementat pentru J2ME. Această tehnologie este totuşi o tehnologie prezentă pe o scară largă de dipozitive, existând o continuă încercare de îmbunătăţire a tehnologiei.

Atuul framework-ului Peer2Me este reprezentat de faptul că foloseste un model peer-to-peer pur în crearea comunicaţiilor în reţelele ad-hoc. Aceasta soluţie elimină problema existenţei unui punct de eşec unic prezentă în tipologiile de tipul Client/Server sau Master/Slave. De asemenea, se utilizează întreaga lăţime de bandă a tipologiei reţelei.

Un alt aspect important al frameworkului este faptul ca încapsulează toate clasele legate de reţea şi prezintă o interfaţă generică prin intermediul căreia se dezvoltă MIDlet-uri. Această soluţie reduce timpul şi codul necesar unui dezvoltator pentru crearea unui MIDlet şi nu necesită cunoştinţe despre tipologia reţelei. Noi module network pot fi adăugate fără a modifica interfaţa MIDlet-ului sau codul acestuia.

Existenţa API-ului FileConnection (JSR-75) permite MIDlet-ului să acceseze sistemul de fişiere local a dispozitivului pe care rulează. Metodele pentru realizarea acestui acces la fisiere sunt încapsulate de framework, acesta prezentatnd MIDlet-ului metode pentru transferul de fişiere sau pentru navigarea în sistemul de fişiere.

Codul framework-ului este bine structurat şi oferă comentarii necesare înţelegerii codului şi denumiri sugestive ale metodelor şi atributelor.

Ca urmare a optimizărilor aduse prin implementarea algoritmului de rutare propus, noul avantaj al framework-ului constă în oferirea suportului pentru realizarea reţelelor scatternet prin implementarea unui protocol avansat de rutare a pachetelor în reţea. Prin adăugarea acestui suport, dispozitivele care nu se gasesc unul în proximitatea celuilalt pot comunica prin intermediul altor dispozitive din cadrul reţelei piconet individuale.

Partea VI

Concluzii şi direcţii de dezvoltare

Capitolul 11. Concluzii şi direcţii de dezvoltare ulterioară

În acest capitol vor fi prezentate concluziile rezultate în procesul de realizare a acestei lucrări de diplomă şi eventualele direcţii de dezvoltare a framework-ului Peer2Me.

Principala concluzie este reprezentată de constatarea făcută pe parcursul procesului de cercetare, şi anume faptul ca există foarte puţine framework-uri open source, cu o reţea pură peer-to-peer dedicate reţelelor ad-hoc formate de dispozitive mobile cu resurse limitate precum telefoanele mobile, şi care să ofere funcţionalitate evoluată pentru aplicaţii colaborative necesare mediilor mobile.

O altă concluzie care poate fi formulată este reprezentată de dificultatea de realizare a unor aplicaţii colaborative pentru telefoane mobile datorită varietăţii telefoanelor mobile prezente pe piaţă la ora actuală.

Cu toate că framework-ul Peer2Me este un framework funcţionabil, există încă funcţionalităţi care pot fi adăugate în viitor sau anumite funcţionalităţii care pot fi optimizate.

11.1 Obiective pe termen scurt

Algoritmul implementat este capabil, la ora actuală, de rutarea unui singur mesaj la un moment dat. Astfel, unul dintre obiectivele pe termen scurt ar consta în adaptarea framework-ului situaţiei reale de funcţionare a unei reţele scatternet, astfel încât să devină posibilă rutarea unui număr mare de mesaje.

Odată cu implementarea acestui algoritm de rutare, devine posibilă dezvoltarea de agenţi mobili. Un agent mobil este o compoziţie de software şi date, capabil să migreze autonom de pe o unitate computaţională pe alta, şi să îşi continue execuţia pe unitatea destinaţie. Pentru realizarea agenţilor mobili, trebuie create obiecte serializabile care sunt trimise apoi la un alt dispozitiv. Pe dispozitivul destinaţie, obiectul trebuie să fie apelat pentru a-şi realiza obiectivul.

11.2 Obiective pe termen îndelungat

Un obiectiv pe termen lung ar fi acela de adăugare a suportului pentru alte tehnologii de reţea. La ora actuală, Peer2Me suportă doar tehnologia Bluetooth, dar alte tehnologii de reţea pot fi integrate relativ uşor.

Pentru a evalua valoarea aplicaţiilor colaborative destinate telefoanelor mobile, aplicaţiile stabile şi uşor de folosit ar trebui distribuite ca open source, cu scop de testare pentru îmbunătăţiri ulterioare a funcţionalităţilor oferite.

Într-o lume în continuă schimbare, telefoanele mobile ocupă un loc tot mai important în modul în care oamenii colaborează şi comunică. În aceste condiţii dezvoltarea de aplicaţii inovatoare dedicate acestui domeniu prezintă un mare interes în randul dezvoltatorilor de software. Trimiterea mesajelor gen „sms” prin intermediul tehnologiei Bluetooth reprezintă de o mare provocare, iar ideea acestei lucrări de diplomă doreşte să dea startul în realizarea acestui proces colaborativ interuman.

Bibliografie

[1] Pew Internet & Americam life project
http://www.pewinternet.org/pdfs/PIP_Mobile.Data.Access.pdf

[2] R.E.Johnson and B.Foote. Designing reusable classes, *Journal of Object-oriented Programming*, 1988.

[3] Steinar A. Hestnes, T. Vatn, Redesign and optimalization of the Peer2Me Framework; A Framework for developing Application supporting mobile collaboration using J2ME, Master's thesis, IDI, NTNU, 2006.

[4] Roy L. Ashok and Dharma P. Agrawal. Next-generation wearable networks, *IEEE Computing Practices Magazine*, pages 31-39, 2003.

[5] Stefano Basagni, Marco Conti, Silvia Giordano, and Ivan Stojmenovic, *Mobile ad hoc Networking.* IEEE, 2004.

[6] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, and Zhichen Xu. Peer-to-peer computing. Tehnical report, HP Laboratories Palo Alto, 2002.

[7] Michael Miller. *Discovering Bluetooth, Learn What You Can Do with Bluetooth Today – and What You'll Be Able to Do Tomorrow*. SYBEX, 2001.

[8] Yaniv Shaked and Avishai Wool. Cracking the bluetooth pin. In *MobiSys 2005Ş Procceding of the 3rd international conference on Mobile systems, applications, and services*, pages 39-50, New York, USA, 2005, ACM Press.

[9] Yi Lu and Serge Vaudenay. Faster correlation attack on bluetooth keystream generator e0. In *Advances in Cryptologyâ CRYPTO 2004: 24th Annual International Cryptology Conference, Santa Barbara, California, USA, 2004*, page 407. Springer Berlin/ Heidelberg, 2004.

[10] E. Vergetis, R. Guerin, S. Sarkar, and J. Rank. Can Bluetooth succeed as a large-scale ad-hoc networking technology? *Selected Areas in Communications, IEEE Journal on*, 2005.

[11] Michael Sars Norum and Carl-Henrik Wolf Lund. *The Peer2Me Framework; A Framework for Mobile Collaboretion on Mobile Phones*, Master's thesis, IDI, NTNU, 2005.

[12] E. Vollset and P. Ezhilchelvan, *Design and Evaluation of an Efficient Reliable Manycast protocol for Ad-hoc Networks*, 2004.
http://www.cs.ncl.ac.uk/research/pubs/trs/papers/838.pdf

[13] C. Carter, S. Yi, P. Ratanchandani, and R. Kravets. Manycast exploring the space between anycast and multicast in ad-hoc networks. In *Proccedings of the 9th annual international conference on Mobile computing and networking*, pages 273-285, ACM Press, 2003.

[14] Sumi Helal. Pervasive java. *IEEE Pervasive Computing, 2002.*

Subliniaza articolul de baza si specificatia J2ME

La carte se pune ISBN, la articole de pe net locatia de unde au fost accesate: link