МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

іНСТИТУТ КОМП’ютерних НАУК та ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

### *Кафедра “Системи автоматизованого проектуванн*



Звіт

до розрахункової роботи

на тему:

Агенти JADE на мобільних пристроях

Виконала

студентка гр. СПКм-12

Столярчук Наталія

Прийняв

Романюк А.Б.

Львів-2014

Розділ 8

АГЕНТИ JADE НА МОБІЛЬНИХ ПРИСТРОЯХ

Запровадження безпровідних мереж, завжди пов’язаних між собою (GPRS, UMTS, WLAN), а також постійне зростання потужності та ресурсів ручних пристроїв, таких як КПК або мобільні телефони, стали очевидною причиною прогресивної інтеграції безпровідного та провідного обладнання. У свою чергу значно зростає необхідність у розробці додатків для застосування частково у стаціонарних мережах, частково у мобільних пристроях. Агенти JADE можна запустити на мобільних пристроях із профілем MIDP завдяки додатку LEAP, розробленому в 2002 році компаніями Motorolla, British Communications, Broadcom Eireann, Siemens AG, Telecom Italia та Пармським університетом як частина проекту LEAP, частково профінансовансованого Європейською комісією (IST 1999-10211). Дана глава зосереджена на цій рисі JADE.

Завдяки додатку LEAP стало також можливим використовувати агенти JADE на платформі Microsoft.NET. У даній книзі, однак, про цю функціональність не йтиметься. Зацікавлені читачі можуть звернутися безпосередньо на сайт JADE або до інструкції для користувача LEAP, яку можна знайти у додатку LEAP.

8.1. Основні обмеження до мобільного обладнання

При розробці агентів для таких мобільних пристроїв як КПК або мобільні телефони, до уваги слід взяти ряд обмежень. Вони пов’язані із обмеженими можливостями самих пристроїв, обмеженими функціями, які підтримує віртуальна машина Java на цих пристроях, а також із типом безпровідних мереж, які використовуються, наприклад GPRS або UMTS. Даний розділ описує ці питання, які в свою чергу слугують основою для архітектурних рішень при розробці додатка LEAP.

8.1.1 Обмеження для апаратного забезпечення

Ручні прилади зазвичай володіють зменшеною продуктивністю з 16-бітними процесорами, і в загальному не перевищують 200 MHz. У них також обмежена пам’ять – від 64 kb у SIM-картках до приблизно 16 Mb в телефонах та 64 Mb в КПК. Серед інших важливих обмежень варто назвати постійну пам’ять, у якій відсутня файлова система та обмежений час роботи батареї. Хоча телефони нового покоління успішно долають ці обмеження, а різниця між можливостями ноутбуків та ПК щодня зменшується, для того, щоб виправдати очікування користувачів, розробники повинні приділити значну увагу пам’яті та продуктивності.

8.1.2. Обмеження JAVA

Віртуальна машина Java, доступна на ручних пристроях, не забезпечує усіх можливостей, доступних на ноутбуках або ПК. Зокрема, характеристики профілю MIDP (єдиного, який зустрічається на більшості мобільних телефонів) не підтримує вхідних зв’язків. Навіть на КПК, де зазвичай доступна віртуальна машина Java, підлаштована під особистий профіль, конфігурацію CDC або старі характеристики персональної Java, можливість відкриття з’єднання для пристрою не завжди гарантована. Причиною для цього зазвичай є використання мобільними операторами брандмауера. Ще одне обмеження характеристики MIDP стосується Java Reflexion, Java Serialization та Collection Framework. Також даний профіль володіє обмеженими можливостями для графіки та інтерфейсу користувача.

8.1.3 Обмеження мережі

JADE забезпечує постійний зв’язок між контейнерами. Кожен контейнер повинен володіти можливістю відкрити та прийняти мережеві з’єднання від і до всіх інших контейнерів платформи у будь-який час. Крім того, як пояснювалося у Розділі 3.6.2.1, усі взаємодії між контейнерами у JADE відбуваються з використанням протоколу RMI, який є не зовсім ефективним, якщо брати до уваги кількість байтів, що передаються мережею. На жаль, безпровідні мережі на кшталт GPRS та UMTS не підтримують подібних функцій оскільки вони характеризуються:

1. Нерівномірна якість зв’язку через області з поганим покриттям, екрановані середовища та потреба вимикати пристрої для збереження потужності батареї;
2. Мінливі IP-адреси – IP-адреса мобільного пристрою не обов’язково залишається тією ж після тимчасового від’єднання;
3. Довгий час очікування мережі та низька пропускна здатність, що коливається від 9.6 kbps (GSM) до 128 kbps (GPRS), 1 Mbps (HSDPA) та далі.

8.2. Додаток LEAP

В кінці 1999 року група лідерів у сфері мобільних телекомунікацій вирішила об’єднати свої зусилля для розробки спільної платформи, яка б підтримувала розробку агентів для мобільних пристроїв сумісних із FIPA. Групу очолювала компанія Motorola, також до неї входили основні виробники мобільних пристроїв - Siemens та Ericsson (представлені Broadcom, центр досліджень Ericsson в Ірландії), та два основних мобільних оператори – British Telecommunications та Telecom India. Результатом такої ініціативи став проект в програмі Єврокомісії IST, запущений на початку 2000 року. Проект назвали LEAP, Lightweight Extensible Agent Plantform, оскільки його основною метою було створення міжплатформного програмного забезпечення, достатньо простого для його використання на пристроях з обмеженими ресурсами, зокрема мобільних телефонах. Система також повинна була стати достатньо розширюваною для забезпечення підвищеної функціональності при її використанні на складніших пристроях.

Команда проекту LEAP зрозуміла, що оптимальним шляхом була не розробка абсолютно нової платформи, а вибір вже існуючої та її пристосування до вимог проекту. Після декількох місяців оцінки проект обрав JADE як основу для LEAP. Такий вибір пояснювали наступними причинами:

* Java 2 Micro Edition на той час фактично стала стандартом для розробки додатків для мобільних клієнтів.
* JADE повністю написаний Java.
* JADE є відкритим джерелом, таким чином надаючи проекту доступ до всіх своїх джерел. Крім того, досі JADE зібрав значну кількість користувачів, які можуть стати хорошою основою для застосування нової простої розширюваної платформи агента.
* Деякі риси JADE, зокрема можливість виконання багатьох завдань одночасно у єдиному потоці Java, добре підходили для пристроїв з обмеженими ресурсами.
* Зрештою, розробка JADE проводилася Telecom Italia, членом проекту LEAP.

Нові впровадження, розроблені проектом LEAP, представили як додаток JADE і відкрили для публічного доступу у 2003 році. Власне, додаток замінив певні частини ядра JADE, формуючи модифіковане робоче середовище, відоме під назвою JADE-LEAP та здатне бути пристосованим для ряду пристроїв, від серверів до мобільних телефонів із Java. Щоб підійти вимогам усіх цих пристроїв, JADE-LEAP може формуватися у трьох різних напрямках, відповідно до трьох видів середовища Java (видань, конфігурацій та профілів), які можна знайти на необхідних пристроях:

* J2SE: для запуску JADE-LEAP на ПК та серверах із фіксованою мережею, що працює на основі JDK1.4 або новішій версії
* pJava: для запуску JADE-LEAP на ручних пристроях, що підтримують J2ME CDC або Personal Java.
* MIDP: для запуску JADE-LEAP на ручних пристроях, що підтримують MIDP1.0 (або пізніші версії), наприклад більшість мобільних телефонів із Java.

Незважаючи на внутрішні відмінності, три версії JADE-LEAP для J2SE та pJava не підтримуються у JADE-LEAP для MIDP, оскільки вони відносяться до класів Java, що не підтримуються MIDP. Версія pJava повністю видаляє усі графічні засоби адміністратора, при цьому зберігаючи усі інтерфейси програмування додатків версії J2SE.

JADE-LEAP для J2SE, pJava та MIDP можна завантажити безпосередньо у бінарній формі із області «завантаження» на веб-сайті JADE. Тим не менш, при розробці JADE-LEAP додатків для MIDP корисно завантажити ресурси JADE разом з додатком LEAP із області «додатків» на веб-сайті JADE. Додаток володіє певними засобами, які можуть стати у нагоді для створення правильного пакету додатків JADE-LEAP для середовища MIDP. Дана тема детально розглянути у Розділі 8.4.

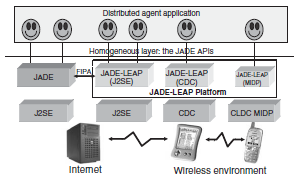


Рис.1. JADE-Leap середовище

8.2.1. JADE і JADE-LEAP

З точки зору розробників додатків і користувачів, JADE-LAEP для J2SE ідентичний JADE з точки зору API, і практично ідентичний під час виконання(див розділ 8.2.1.1).Розробники можуть використовувати своїх агентів JADE на JADE-LEAP і навпаки без жодних змін. Немає необхідності для JADE-LEAP пояснень чи API документації, так як вони передбачені для JADE, можуть використовуватися і для JADE-LEAP.

Та потрібно пам’ятати, що JADE контейнери та контейнери JADE-LEAP не можуть бути змішані на одній платформі. Це головний стимул для використання J2SE версії JADE-LEAP. Без цього було б неможливо розгорнути єдину платформу JADE-LEAP, яка породжує обидві мережі, бездротову і дротову.

Як показано на малюнку 8.1, JADE-LEAP платформа і JADE платформа можуть комунікувати як описано у FIPA, наприклад, використовуючи HTTP MessageTransportProtocol.

8.2.1.1. Відмінності між JADE і JADE-LEAP для J2SE

Як було згадано раніше, JADE-LEAP для J2SE забезпечує ідентичне API таке як у JADE. Проте, є кілька відмінностей у термінах під час виконання додатку.

**Jar файли**: JadeLeap Jar файл, який виробляється під час виконання JADE-LEAP для J2SE також включає класи, пов’язані як і з інструментами управління(вони включені в JADE у jadeTools.jar файл) так і з двома готовими MTPs для HTTP i IIOP(що в JADE відповідно включені у http.jar ті iiop.jar файли).

**Специфікація bootstrap агентів у командній стрічці**: При запуску JADE з командної стрічки, агенти, які повинні бути активовані під час завантаження, повинні бути розділені пробілами(‘ ’). З іншого боку, крапка з комою (‘;’) використовується як розділювач при запуску JADE-LEAP. Наприклад, команда для JADE

java jade.Boot -gui john:myPackage.MyClass peter:myPackage.MyClass

заміняється на

java jade.Boot -gui john:myPackage.MyClass;peter:myPackage.MyClass

коли запускається JADE-LEAP .

**Аргументи агента**: Аргументи агента JADE розділюються за допомогою пробілів (‘ ’), як і в командній стрічці, так і в управлінні GUI; коли кома (‘,’) використовується як розділювач для JADE-LEAP. Наприклад, JADE команда

java jade.Boot -gui john:myPackage.MyClass(arg1 arg2)

заміняється на

java jade.Boot -gui john:myPackage.MyClass(arg1,arg2)

коли завантажується JADE-LEAP. У JADE-LEAP аргументи агента не можуть містити пробілів.

**Властивості командної стрічки**:

* Якщо JADE-LEAP має бути запущена без опцій і тільки зі специфікацією bootstrap агентів, тоді опція –agent має бути використана, так як в командній стрічці Java jade.Boot xxxx значення xxxx інтерпритується як ім’я файлу конфігурації. Наприклад,

java jade.Boot john:MyClass не буде працювати

java jade.Boot –agents john:MyClass має бути використана

* -nomobility i –dump опції не присутні в JADE-LEAP.
* LAEP IMTP може бути сконфігуровано при специфікації номера опції, які відсутні при запуску JADE

8.2.2 LEAP IMTP

Хоча JADE та JADE-LEAP майже ідентичні з точки зору зовнішнього бачення, вони досить відмінні внутрішньо. Одна з найбільш серйозних змін є y IMTP, тобто тієї частини платформи, що замається взаємодією контейнер-до-конейнер взаємодії, як описано в розділі 3.6.2, вона зовсім різна.

Нормальний JADE IMTP оснований на JAVA RMI, а це не підходить для мобільних пристроїв. Тому JADE-LEAP використовує альтернативний IMTP на основі власного протоколу JICP(JADE Inter Container Protocol). Різниця між двома протоколами (RMI i JICP) є основною причиною, чому контейнер JADE-LEAP не може зареєструватися з головним контейнером JADE і навпаки.

LEAP IMTP складається з так званого “Command Dispetcher” і одного або декількох Internal Communication Peers(ICPs). Він відповідальний за:

* серіалізацію і десеріалізацію горизонтальних команд JADE (див . 7.1.3.2), якими обмінюють ся контейнери;
* маршрутизацію серілізованих команд з локального контейнера до відповідного ICP (відповідно до адреси віддаленого контейнера) для передачі по мережі; і
* проходження команд, отриманих за допомогою основних ICP до локальних контейнерів.

ICP відповідально за відправку і отримання серіалізованих команд(наприклад, рядок послідовностей байтів) по мережі, використовуючи наданий протокол. Рисунок 8.2 показує головні компоненти LEAP IMPT.

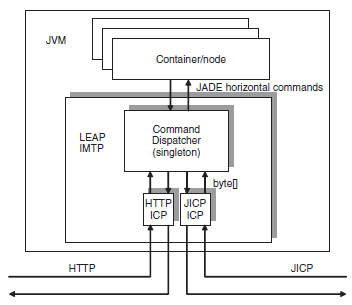


Рис. 8.2

Доповнення LEAP включає в себе три ICP реалізації: JICP-based ICP за замовчуванням реалізується класом jade.imtp.leap.JICP.JICPPeer, SSL версія JICP ICP реалізована класом jade.imtp.leap.JICP.JICPSPeer , і HTTP-based ICP реалізований класом jade.imtp.leap.http.HTTPPeer . Це означає, що JADE-LEAP платформа може використовувати JICP, JICP для SSL, чи HTTP для контейнер-до-контейнер протокол взаємодії.

Потрібно зазначити, що оскільки більше одного ICP може бути активовано у Command Dispatcher, всі контейнери, що знаходяться у місцевому JVM і поділяють Command Dispetcher, тому він може бути доступний за кількома адресами. Наприклад, на рисунку 8.2 контейнери доступні з іншого контейнера за адресою з типом jicp://… і за іншим типом адреми http://...

8.2.2.1 Кофігурація LEAP IMTP

Основний конфігураційний атрибут затверджений LEAP IMTP це icps, який визначає який ICP активувати в даному контейнері. Значення атрибуту – це список ICP, розділених крапка з комою (‘;’). Наприклад:

java jade.Boot -gui -icps

jade.imtp.leap.JICP.JICPPeer;jade.imtp.leap. http.HTTPPeer(2000)

Вищезгадана команда розпочинає роботу головного контейнера JADE-LEAP з двома ICP; перший використовує JICP протокол і очікує на вхідні горизонтальні команди на порт 1099 за замовчуванням(для сумісності з JADE); інший використовує HTTP протокол і очікує горизонтальних команд на порт 2000. Значення за замовчуванням для icps опції це одиночний JICP ICP, що використовує порт 1099 за замовчуванням.

Отже, командна стрічка:

java jade.Boot –gui

замінюється на

java jade.Boot -gui -icps jade.imtp.leap.JICP.JICPPeer

При використанні HTTP протоколу(наприклад, активація HTTPeer ICP) –proto http опція має бути специфікована на периферійних контейнерах в додаток до –host i –port атрибутів.

8.3 РЕЖИМ ВИКОНАННЯ З РОЗДІЛЕНИМ КОНТЕЙНЕРОМ

Зараз має бути зрозуміло, що агенти JADE вимагають переважного часу виконання, що забезпечить їм необхідну функціональність, щоб жити і комунікувати. Переважно час виконання JADE завжди реалізується контейнерами, але LEAP доповнення надає альтернативний спосіб реалізації, відомий як розподілене виконання і спеціально призначене для задоволення потреб мобільних пристроїв.

При запуску JADE на виконання, використовуючи розділений режим виконання, користувач не використовує звичайний контейнер, а дуже тонкий шар під назвою front-end. Front-end забезпечує агентів точно такими ж особливостями, як і контейнер, але реалізовує лише невелику підмножину з них безпосередньо, в той же час делегуючи інших на віддалений процес під назвою back-end. Якщо front-end виглядає як звичайний контейнер з точки зору агентів, що задіяні на ньому, back-end, в свою чергу, виглядає як звичайний контейнер з точки зору інших контейнерів на платформі, включаючи основний контейнер. Результатом є те, що об’єднання front-end та back-end формують контейнер, який розділений на дві частини, утворюючи при цьому термін розділений режим виконання. Front-end та Back-end комуні кують через з’єднання, як показано на рисунку 8.3.

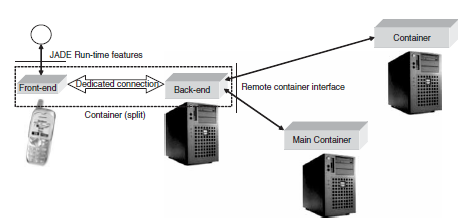


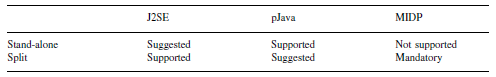
Рис. 8.3.

Розділений режим виконання особливо підходить для обмежених в ресурсах і бездротових пристроїв так, як:

* Front-end, який працює в пристрої, значно легший за повний контейнер.
* Початкова фаза завантаження набагато швидша так, як всі комунікації з головного контейнера, необхідного, щоб приєднатися до платформи виконуються back-endом і тому не проводяться по бездротовому з’єднанню.
* Використання бездротової лінії оптимізовано.
* Front-end та back-end додають склад і відсилають механізм, щоб зробити можливі втрати прозорими для додатку.
* Тобто, якщо зв’язок між front-end та back-end спадає, повідомлення до і від агентів, що знаходяться на front-end буферуються на front-end і back-end. Як тільки зв’язок налагоджується, буферизоване повідомлення відправляється до призначених отримувачів.
* IP адреса мобільного приладу ніколи не може побачити інший контейнер на платформі, так як вона завжди взаємодіють з back-end. Це можна навіть змінити без впливу на додаток.

В таблиці 8.1 підсумовано, як два режими виконання(звичайний stand-alone та розділений) у різних середовищах пристосованих до JADE-LEAP. Треба звернути увагу на те, що розділений режим виконання є обов’язковий у MIDP і бажано його застосовувати у pJAVA. Також важливо зазначити, що розробники агентів не повинні знати чи їх агенти будуть виконуватися на stand-alone чи front-end split контейнері, адже API, які вони забезпечують однакові.

Таблиця 8.1. Режими виконання JADE-LEAP



Однак, є три проблеми, які розробники повинні усвідомлювати:

1. Неможливо виконати головний контейнер, використовуючи розділений режим виконання.
2. jade.core.Runtime клас і класи у jade.wrapper пакеті, описані в розділі 5.6, розроблені для управління stand-alone контейнером. Зовнішні застосунки можуть управляти розділеним контейнером за допомогою мінімальних процесів інтерфейсу, що забезпечує jade.core.MicroRuntime клас(див. 8.5.3). Цей клас доступний лише при роботі з JADE-LEAP, так як розділений режим виконання відсутній у JADE.
3. Мобільність агентів і клонування не передбачено у розділеному контейнері.

При запуску JADE-LEAP на PersonalJava/CDC пристрій і засоби мобільності потрібні, рекомендовано спробувати розділений режим виконання та механізм завантаження динамічної поведінки, що підтримується класом jade.core.behaviours.LoaderBehaviour .Тільки якщо це не підходить для потреб за стосунку, потрібно використовувати stand-alone режим виконання.

Висновок:

Отже, під час виконання розрахункової роботи, я ознайомилася з додатком JADE-LEAР, який призначений для мобільних пристроїв. Розглянула основні відмінності між JADE та JADE-LEAP.