УДК 004.4'2

Анісімов А.В. проф. доктор ф.м., Глибовець А.М. ст.викл., Жаб'юк В.Я.

Основні архітектурні принципи побудови програмних систем реалізації мобільних агентів

В статті розглянуто основні архітектурні принципи побудови програмних систем реалізації мобільних агентів. Проведено порівняльний аналіз найбільш відомих платформ підтримки обчислювальних мобільних агентів. Визначено базові вимоги до архітектури програмних систем такого типу.

Ключові слова: архітектурні принципи, мобільні агенти, платформи мобільних агентів.

E-mail: ava@unicyb.kiev.ua

A.V. Anisimov professor, A.M. Glybovets senior lecturer, V.Y. Zhabyuk

Main Architectural Principles for Mobile Agents

Major architectural principles of mobile agents design are investigated. Comparative analysis of popular mobility support platforms is performed. Basic requirements to such architectures are defined.

Key Words: Architectural principles, mobile agents, Mobile agents platforms.

Вступ

Існують певні причини зацікавленості мобільними агентами (МА) за останні роки. Ранні дослідження МА були важливими, але не оцінені адекватно, поки ставилися в один ряд із дослідженнями складних програмних комплексів реального часу. МА можуть працювати над вирішенням проблеми латентності, пропускної спроможності програм клієнт-сервер та зменшення вразливості мережного з'єднання. WWW та Інтернет використовують МА для підвищення інтелектуальності та доступності своїх сервісів для непрофесійних користувачів. МА також можуть забезпечувати єдину загальну структуру, у якій інформаційно-орієнтовані прикладні програми можуть ефективно застосовуватися та легко поширюватися.

Класифікація агентів на основному рівні поділяє агентів на три категорії: біологічні, автоматизовані (роботи) та обчислювальні агенти [1].

Ми зосереджуємо свою увагу на обчислювальних агентах. Існує декілька загальних характеристик, які входять до визначення такого МА. Як мінімум МА є автономною, самостійною й визначеною комп'ютерною програмою, яка здатна рухатися в мережі, взаємодіяти в середовищі та діяти від імені користувача чи іншої особи. Можна виділити багато прикладних задач, для розв'язку яких ефективним є використання МА. Серед них виділяють: управління та супровід; електронна комерція; мобільні пристрої, мобільні користувачі та мобільне обчислювання, пошук інформації та повернення; безпечне посередництво; послуги телекомунікаційних мереж; прикладні програми технологічного процесу й програмне забезпечення для спільної роботи групи людей (groupware); одночасна (паралельна) обробка.

Наприклад, у великих мережах моніторинг, виявлення помилок та дистанційне інсталювання програмних компонентів є досить складним та включає в себе велику кількість реєстраційних даних. Неможливо створити вбудовану діагностичну програму для кожної окремої цілі, але можна використовувати МА для моніторингу системи (Perpetuum Mobile Procura [2]) та звертати увагу адміністратора системи на проблемні місця. Агенти можуть бути досвідченими асистентами для демонстрації інформаційних систем, перемішуватися в місця збереження даних замість переміщення

всієї інформації через мережу та пошуку необхідної інформації, даючи людям можливість використовувати інформацію опосередковано й ефективно (система Stormcast [3]). МА здатні клонувати себе в мережі, це робить їх придатними для завдань одночасної обробки.

Саме через різноманітні підходи до дослідження технології агента виникла необхідність створення єдиної організації стандартизації цих досліджень. Фонд інтелектуальних фізичних агентів (FIPA) був створений у 1996 році з метою заповнення цієї прогалини. Пізніше з'явилася група управління об'єктами (OMG), яка є особливо активною в поданні ініціатив та стандартів функціональної сумісності й безпеки. Окрім FIPA і OMG існує ще декілька організацій зі стандартизації, які працюють у напрямку створення ініціатив і стандартів. Так, IETF (Інтернет-група особливого призначення) приділяє особливу увагу технологіям Інтернет-агентів. W3C займається веб-мовами та онтологією мобільних агентів. Серед інших дослідницьких груп та офіційних форумів, які поширюють (промотують) технології мобільних агентів виділяють Agent Society, NIST та Agent InteropWorking Group.

Усе це зумовило розглянути та проаналізувати існуючі базові архітектурні принципи побудови програмних систем реалізації мобільних агентів.

Системи підтримки МА

Сьогодні існує багато систем створення спеціалізованих МА, деякі з них усе ще знаходяться у фазі дослідження, деякі є комерційними системами. Серед них можна виділити: Bee-agent [4], Aglets [5], Hive [6], AgentTcl [7] та Telescript [8]. Розглянемо далі системи, які демонструють певні особливості архітектурних підходів до програмної реалізації і є типовими для свого класу.

ARA. Система Агента віддаленої дії (ARA) [9] є платформою для мобільних агентів, що створюється в університеті Кайзерслаутери (Kaiserslautern). Мобільні агенти Ara програмуються в Tcl, Java і C/C++. Для всіх трьох мов Ara забезпечує команду *руху (go)*, яка автоматично фіксує повністю стан агента, переносить його до потрібної машини, і відновлює виконання агента саме в точці *руху (go)*. Ara також дозволяє агенту робити контрольну перевірку (checkpoint) його поточного внутрішнього стану в будь-який час.

На відміну від інших подібних систем, Ara має багато потоків (ниток, відгалужень): сервер агента та інтерпретатори алгоритмічної мови працюють у середовищі єдиного процесу Unix. Хоча цей підхід ускладнює виконання, він має істотні комунікаційні переваги, за рахунок використання невеликого інтерпретатора. Коли з'являється новий агент, він просто починає виконання в новому потоці. У випадку потреби зв'язатися з іншим, він просто переміщує структуру повідомлення до потрібного агента, заміняючи використання міжкомунікаційних процесів.

Група Ara завершує імплементацію початкових механізмів безпеки. Кодування агента проводить програміст, а його параметри та загальний дозвіл на користування ресурсами визначаються його власником (користувачем). Кожна машина має один або більше віртуальних просторів, створених агентами. Мігруючий агент повинен переміститись в певне місце. Коли він прибуває на це місце, функція прийняття відмовляє агенту в доступі, або ж надає набір дозволів, грунтуючись на його криптографічних посвідченнях (даних).

D'Agents. D'Agents, колись відомий як Agent Tcl [10], є системою мобільного агента, створеною в Дартмутському коледжі в Ганновері. Він базується на скриптовій мові Tcl для керування агентів, а Safe Tcl використовується для гарантування безпеки. Подібно до Ara, D'Agents забезпечує команду *руху (go)*, автоматично фіксує й відновлює повний стан мігруючого агента. На відміну від Ara, тільки сервер D'Agent має багато потоків (ниток), а кожен агент виконується в окремому процесі, який значно спрощує імплементацію, але ускладнює внутрішню комунікацію.

Сервер D'Agent використовує криптографію (шифр) публічного ключа, щоб засвідчити (ідентифікувати) власника агента, що прибув. Стаціонарні агенти ресурсного менеджера призначають права доступу агентові, що базувалися на автентифікації й перевагах адміністратора. Модулі зі специфічними мовами встановлюють права доступу та запобігають виникненню

порушення (до прикладу, доступ до файлової системи) чи знищення агента при настанні порушення (наприклад, коли загальний час CPU закінчується). Кожен менеджер ресурсів асоціюється зі специфічним (певним) ресурсом, до прикладу, файловою системою. Менеджери ресурсів можуть бути як завгодно складні, але за замовчанням менеджери просто асоціюють список прав доступу з кожним власником.

На відміну від Ara, з більшістю менеджерів ресурсів не консультуються при появі агента. Звернення до них виникає тільки коли агент (1) намагається отримати доступ до відповідного ресурсу або (2) явно вимагає право специфічного доступу. У цьому випадку менеджер ресурсу пересилає всі відповідні права доступу до примусового модуля. і D'Agents поводиться так само як Ara, надаючи права доступу in short wrapper functions навколо функцій доступу до ресурсу. D'Agents використовується в деяких прикладних програмах пошуку інформації.

Aglet. Aglets створений в IBM (Японія). Агенти в Aglets є об'єктами Java, які можуть рухатися від одного інтернетівського хоста до іншого. Тобто агент, який виконується на одному хості, може раптово призупинити виконання, переміститись на віддалений хост та продовжити виконання вже там на ньому. Коли агент рухається, він переносить із собою свій програмний код і свої дані.

Замість надання прототипу (простого) *go* Ara і D'Agents, Aglets використовує варіант моделі Tacoma, у якій виконання агента розпочинається знову з початкової точки після кожної міграції. Зокрема, Aglets використовує модель, орієнтовану на події. Коли агент хоче мігрувати (переміститись), він застосовує метод *відправки*. Система Aglets застосовує *onDispatching* метод агента, який виконує специфічну чистку, знищує потоки (нитки) агента, упорядковує код агента й стан об'єкта, і відправляє код та стан об'єкта до нової машини. На новій машині система застосовує *onArrival* метод агента, який виконує специфічну ініціалізацію, а потім застосовує метод запуску агента для його повторного виконання. Aglets містить: простий засіб постійного зв'язку, який дозволяє агентові записувати його код і стан об'єкта до зовнішньої пам'яті, і тимчасово "вимикати" себе; програми-посередники (proxies), які діють як представники Aglets та забезпечують прозорість розташування, обслуговування пошуку для знаходження мобільних (тих, що рухаються) Aglets, та низку засобів для комунікації між агентами шляхом передачі повідомлень.

Telescript. Telescript [11], створений General Magic на початку 1990-их років як перша система, спроектована спеціально для підтримки мобільних агентів у комерційних прикладних програмах, включає, орієнтовану на об'єкт безпечного типу, мову для програмування агента. Сервери Telescript, яких ще називають *місцями*, надають послуги, зазвичай, інсталювання стаціонарних агентів для спілкування з гостьовими агентами. Агенти використовують примітивну команду *руху (go)* для абсолютної міграції до місць, указаних при використанні імен хостів, базованих на DNS. Система фіксує режим виконання на рівні потоку, таким чином, агент відновлює операцію негайно після *руху (go)*. Відносна міграція також можлива при використанні простої команди *зустрічі*. Співвіднесені агенти можуть використовувати методи один одного для комунікації. Засіб, що повідомляє про подію, також є доступним. Telescript не був комерційно успішним тому, що використання його вимагало вивчення програмістами цілком нової мови. У 1997 році підтримуюча компанія, General Magic Inc відклала проект Telescript і почала роботу над новою системою, що базується на Java мові, під назвою Odyssey [12], яка використовує ту ж саму архітектуру.

Ajanta. Ajanta (Проект дослідження мобільних агентів) – система програмування мобільного агента, створена в університеті Міннесоти. Перша пробна версія Ajanta була представлена 1999 році, а в 2003 була випущена друга комерційна версія Ajanta. Агентська система Ajanta побудована на базі Java, та створена на основі агентських платформ мобільних серверів та мобільних агентів.

У Ajanta агент виконується в ізольованій захищеній області для запобігання несанкціонованого втручання з боку інших агентів. Сервер захищає свої ресурси, розміщуючи їх у проксі об'єкти, які створені динамічними й налаштовані (адаптовані) під кожного окремого клієнта агентів.

Аналогічний механізм може дозволити безпечну комунікацію посередника через застосування методу активізації. Комунікація через мережу є також можливою при застосуванні віддаленого методу активізації. Аутентифіковані контрольні функції дозволяють прикладним програмам повторно викликати або закінчувати (зупиняти) своїх віддалених агентів у будь-який час. Ajanta також працює над проблемою захисту стану (статичної інформації) агента від небажаних (злоякісних) серверів. Вона надає криптографічні механізми, які дозволяють власнику агента захистити частину стану агента та виявляти будь-яке подальше втручання.

Voyager. Агентська система Voyager, що базується на мові Java, створена Object Space, забезпечує зручний частково прозорий спосіб взаємодії з віддаленими об'єктами (через проксі об'єкт із «віртуальним» посиланням) і спосіб для об'єктів переміщуватися від одного до іншого хоста [13]. Коли об'єкт переміщується, він залишає за собою «форвардний» об'єкт, який пересилає будь-які повідомлення на нове місце. Об'єкти «агенти», на відміну від інших об'єктів, можуть самостійно переміщувати самих себе, застосовуючи метод *moveTo()*. Voyager переміщає код і дані агента, але не сам потік, до нового розташування, і застосовує там бажаний метод. Voyager також дозволяє об'єктам і агентам стати постійними за допомогою точного *saveNow()*, підтримує групову комунікацію (мульти) і надає послугу федеративної директорії. Voyager забезпечує ті ж основні механізми безпеки, як і інші Java-системи.

Tacoma. Tacoma – спільний проект норвезького Університету Тромсо та Університету Корнелл [14]. Проект зосереджується на підтримці операційних систем для МА і на тому, як МА може допомогти у вирішенні проблем, які традиційно стосуються операційних систем. Tacoma пропонує абстракції, наприклад, брифкейс, папку та файл, для розширення операційних систем із метою забезпечення механізмів передачі даних агенту-посереднику та агентам, що складають розклад. Це підтримує й синхронну, і несинхронну комунікацію. Механізм альтернативної комунікації – використання файлів, які є статичними архівами для загальної інформації. Агенти можуть зберігати специфічні програмні дані у файлах, до яких інші агенти потім можуть отримати доступ. На відміну від Aga і D'Agents, Tacoma не надає автоматичних пристроїв збору даних. Натомість, коли агент хоче переміститись на нову машину, він створює папку, у яку вміщує свій код та будь-яку необхідну статичну інформацію. Папка надсилається до нової машини, де запускається необхідне для виконання програми середовище, а потім звертається до відомої точки входу в межах коду агента для відновлення виконання агента. Для стійкості до помилок Tacoma використовує систему перевірки та використовує захисних агентів для простеження міграції мобільних агентів.

Concordia. Concordia була створена компанією Mitsubishi Electric [15]. Вона написана на Java і зосереджує основну увагу на безпеці та надійності. Вона переміщує код та дані агента, але не стан потоку інформації від однієї машини до іншої. Подібно до інших систем, агенти Concordia прив'язані до маршруту відвідування, який може бути модифікований агентом під час виконання маршруту. Агенти, події та повідомлення можуть бути поставлені в чергу, якщо віддалений вузол зараз недоступний. Агенти ретельно зберігаються в постійній пам'яті перед переміщенням із сайту та після прибуття на новий сайт для того, щоб запобігти втраті агента при руйнації машини. Агенти захищені від втручання шляхом кодування, поки вони перебувають у процесі передачі чи зберігаються на диску; хости агентів захищені від злісних злоякісних агентів шляхом криптографічної автентифікації власника агента та списків контролю доступу, які охороняють кожний ресурс.

Jumping Beans. Комп'ютери, які хочуть прийняти мобільних агентів, використовують Jumping Beans [16] агенцію, яка асоціюється з доменом Jumping Beans. Кожен домен має центральний сервер, який проводить перевірку агенцій, що приєднуються до домена. Мобільні агенти рухаються від агенції до агенції в межах домена, і агенти можуть відправити повідомлення іншим агентам у домені. Обидва

механізми застосовуються шляхом проходження через сервер домена. Таким чином, сервер стає центральним пунктом для дослідження, управління та перевірки агентів. Він також стає центральним місцем помилок чи складних ситуацій/тупиків у виконанні, хоча й компанія планує створити великомасштабні сервери, які б функціонували на паралельних машинах. Інший підхід до універсальності (розширення масштабів) – створити багато маленьких доменів, кожен з яких мав би свій власний сервер. В існуючій версії агенти не можуть мігрувати між доменами, але компанія планує вможливити міграцію між доменами в майбутніх версіях. Безпека й надійність – ключові аспекти Jumping Beans. Криптографія публічного ключа використовується для перевірки відповідності агенцій та серверів, а списки контролю доступу використовуються для контролю доступу агента до ресурсів, базованому на дозволах, наданих власнику агента.

Obliq. Завданням Obliq є збір названих матеріалів, які містять методи, псевдоніми та цінності [17]. Об'єкт може бути створений на дистанційному вузлі, клонований на дистанційному вузлі чи перемішений у комбінації із клонуванням та перенаправленням. Застосування мобільних агентів тут є дуже простим. Агент складається із процедури визначення користувача, яка сприймає брифкейс як аргумент. Брифкейс містить завдання Obliq, які необхідно виконати під час цієї процедури. Агент перемішується шляхом надсилання своєї процедури та наявного брифкейсу до цільової машини, яка розпочинає процедуру завершення виконання агента.

Агент Bee. Бджолиний агент [18], створений Центром лабораторних корпоративних досліджень та розвитку комп'ютерних та мережених систем корпорації TOSHIBA у 1999 році, є ще одним типом архітектури агентської системи. На відміну від інших систем, які лише частково використовують мобільних агентів, Агент Bee повністю "Agentifies" комунікації, які відбуваються між прикладними програмами. Прикладні програми стають агентами; усі повідомлення передаються агентами. Агент Bee дозволяє розробникам створювати гнучкі відкриті дистрибутивні системи, які оптимально використовують існуючі прикладні програми, а також здатний працювати в Java 2 середовищі.

Hive. Система *Hive* є децентралізованою системою для створення прикладних програм за допомогою системних ресурсів локальної мережі. Вона застосовується в роботі мережі "Things that Think" (Речі, що думають) [19], яка включає обчислення та комунікацію щоденних об'єктів та утиліти (appliances). Система *Hive* складається із трьох компонентів: щільникові вузли в децентралізованій мережі, приховані локальні ресурси й обчислення, що здійснюється агентами. Її мобільність є в багатьох випадках повторним застосуванням таких систем як Telescript, AgentTcl, Aglets, та Mole. Проте планується здійснити дослідження для створення простої версії мобільності для складних проблем.

Висновки

Сучасний стан використання агентних підходів до оптимального розв'язку прикладних задач засвідчує існування проблеми побудови програмної архітектури, яка дозволяє ефективно існувати й перемішуватись агентам у розподіленому й гетерогенному середовищі. Ця проблема вирішується за допомогою виділення спеціалізованої *агентної платформи (АП)*, яка контролює та управляє агентною спільнотою [20].

Для вирішення проблем інтероперабельності пропонується дотримуватися таких стандартів при розробці АП. Платформа має бути динамічною структурою, що забезпечує інтеграцію систем, які базуються на різних апаратних і програмних архітектурах. Вона повинна включати в себе різні протоколи зв'язку, для забезпечення спілкування різним агентам, які належать до різних платформ. Звісно, для реалізації можливості незалежної роботи АП має бути автономною [21].

FIPA рекомендує включення наступних фундаментальних модулів у агентну платформу:

- Agent Communication Channel (ACC) - канал спілкування агентів, цей модуль є унікальним і обов'язковим для кожної агентної платформи. Цей модуль відіграє роль маршрутизатора повідомлень.

- Agent Management System (AMS) – система керування агентами. Цей модуль контролює й керує агентною платформою й активністю агентів. Він керує процесом створення, реєстрацією й знищенням реєстрації кожного агента платформи. Також AMS містить список усіх існуючих агентів, що є ідентифікуються по їх Global Unique Identifier (GUID – глобальний унікальний ідентифікатор). Цей список надається у вигляді сервісу, кожному агенту платформи.

- Directory Facilitator (DF). Цей модуль на відміну від списку, що надається ANS містить список агентів домену. Усі агенти, що є зареєстрованими в ньому формують домен агентів. Кожний DF сервіс керує відповідним агентним доменом.

Розробка АП, що задовольняє зазначеним вимогам є актуальною задачею, яка чекає свого розв'язку.

Література

1. Гороховський С.С. Агентні технології: спроба критичного огляду. // Наукові записки. Національний університет "Києво-Могилянська Академія". - Т. 18: Спеціальний випуск: НаУКМА. - К., 2000.–с.391-395.
2. Perpetuum, "Perpetuum Mobile Procura Project," http://www.sce.carleton.ca/netmanage/perpetuum.shtml, 2002.
3. Stormcast 5.0. http://www.cs.uit.no/DOS/StormCast/, 2002.
4. Toshiba, "Multi-Agent Framework for 100% Pure Agent System ," http://www2.toshiba.co.jp/rdc/beegent/index.htm, 2005.
5. IBM, "Aglets," http://www.trl.ibm.co.jp/aglets/, 2001.
6. N. Minar, M. Gray, O. Roup, R. Krikorian, and P. Maes. "Hive: distributed Agents for Networking Things." IEEE concurrency. Vol. 8. No. 2. pp. 24-33, 2000.
7. R. Gray, "AgentTCL: A Flexible and Secure Mobile-agent System," PhD Thesis, Dartmouth College, 1997.
8. J.E. White, "Telescript Technology: Mobile Agents." Software Agents, J. Bradshaw, ed., pp. 437–472, 1996.
9. H. Peine and T. Stolpmann, "The Architecture of the Ara Platform for Mobile Agents," Proceedings of the 1st Internatioanl Workshop on Mobile Agents (MA'97), pp. 50-61, 1997.
10. R. Gray, D. Kotz, G. Cybenko, and D. Rus. "D'Agents: Security in a Multiple-Language, Mobile-Agent System." Mobile Agents and Security, G. Vigna, editor, pp. 154-187, 1998.
11. J.E. White, "Mobile Agents," General Magic White Paper., http://www.genmagic.com/agents, 1996.
12. Odyssey, http://www.genmagic.com/, 2001.
13. ObjectSpace, Inc. "ObjectSpace Voyager Core Package Technical Overview," Technical Report, ObjectSpace, Inc., 1997, http://www.objectspace.com/.
14. D. Johansen, R. Renesse, and F. Schneider. "Operating System Support for Mobile Agents," Proceedings of the 5th IEEE Workshop on Hot Topics in Operating Systems, pp. 42-45, 1995.
15. T. Walsh, N. Paciorek, and D. Wong, "Security and Reliability in Concordia," Mobility: processes, computers, and agents, pp. 524 - 534, 1999.
16. "Jumping Beans White Paper," Ad Astra Engineering, Inc., 1998, http://www.JumpingBeans.com.
17. M. Brown and M. Najork, "Distributed Active Objects," Dr. Dobb's Journal, Vol. 22, No. 3, pp. 34-41, 1997.
18. "FIPA Interaction Protocol Library Specifications," http://www.fipa.org/specs/fipa00025/PC00025C.html, 2002.
19. Things That Think, http://www.media.mit.edu/tt/, 2002.
20. Н. Глибовец Использование агентных технологий в системах дистанционного образования, Журнал "Управляющие системы и машины", 2002. - №6.- С. 69-76.
21. Глибовець М.М. Інтелектуалізація навчання в телекомунікаційних системах дистанційної освіти // Вісник Київського університету. Серія: Фізико-математичні науки. -2002. - Випуск №3. - С. 203-211.

Надійшла до друку 01.09.2008 р.