УДК 681.3, 621.3

Ю.В. Морозов, І.І. Пастернак
Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин.

КЛАСИФІКАЦІЯ ЗАСОБІВ МОДУЛЬНОЇ ВЗАЄМОДІЇ МІЖ КЛІЄНТОМ І СЕРВЕРОМ

Запропоновано варіант класифікації засобів взаємодії між клієнтською і серверною програмами.

Ключові слова: класифікація, клієнт, сервер.

The variant classification of interaction between client and server software describe.

Key words: classification, client, server.

Вступ

Зі зростанням поширеності Інтернету робота з віддаленою інформацією стала звичним явищем. Технологія клієнт-сервер широко застосовується під час роботи з віддаленою інформацією в мережі. Сьогодні, з розвитком глобальної мережі Інтернет, ця технологія все частіше зацікавлює розробників програмного забезпечення, оскільки в світі нагромаджено величезну кількість інформації з різноманітних питань. Дуже важливо правильно уявляти, хто або що розглядається як "клієнт". Можна говорити про клієнтський комп'ютер, з якого відбувається звернення до інших комп'ютерів. Можна говорити про клієнтське та серверне програмне забезпечення. Нарешті, можна говорити про людей, які бажають за допомогою відповідного програмного та апаратного забезпечення отримати доступ до тієї чи іншої інформації.

Загальноприйнятим є визначення, що клієнти та сервери – це передусім програмні модулі. Найчастіше вони є на різних комп'ютерах, але бувають ситуації, коли обидві програми – і клієнтська, і серверна, фізично розміщуються на одному комп'ютері; в такій ситуації сервер часто називається локальним. Модель клієнт-серверної взаємодії визначається передусім розподілом обов'язків між клієнтом та сервером.

Сьогодні не існує загальноприйнятої класифікації методів взаємодії клієнта з сервером. Тому введення такої класифікації дасть змогу аргументовано вибирати конкретні методи.

Огляд літературних джерел

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосувань і передбачає взаємодію та обмін даними між ними [4–6]. Вона передбачає такі основні компоненти:

- набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
- набір клієнтів, які використовують сервіси, що надають сервери;
- мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більше ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

У нашому випадку технології побудови клієнтського програмного забезпечення ґрунтуються на використанні різних типів протоколів. Одним із них є протокол TCP/IP, який використовує сокети [1–3]. Сокети, використовувані протоколом TCP/IP, високостандартизировані і широко-

доступні. Але програмування з застосуванням сокетів програмісти розглядають як занадто низькорівневі. Саме необхідність програмування на низькому рівні перешкоджає продуктивному написанню стійких розподілених додатків. Іншим типом протоколу є протокол віддаленого виклику процедур RPC (Remote Procedure Call). Але протокол віддаленого виклику процедур RPC є доволі складним, і до того ж відомо велику кількість його різновидів. А також є доволі популярними такі протоколи високого рівня, як CORBA (Common Object Request Broker Architecture – архітектура посередника об'єктних запитів), RMI (Remote Method Invocation – технологія віддаленого виклику методів) і розподілена модель компонентних об'єктів DCOM (Distributed Component Object Model). Ці протоколи усе ще складні і для організації їх роботи потрібно наявність спеціального середовища як на стороні сервера, так і на стороні клієнта. Їм притаманні також і інші недоліки. Наприклад, під час використання цих протоколів можливе виникнення проблем при проходженні пакетів даних через брандмауер (системи мережного захисту). Проте, один протокол набув повсюдного поширення. Це протокол передачі гіпертекстових файлів HTTP (Hypertext Transfer Protocol) [7–8]. Саме через повсюдне поширення протоколу HTTP, компанії Microsoft і іншим виробникам мережного програмного забезпечення довелося розробити новий протокол, що одержав назву SOAP (Simple Object Access Protocol – простий протоколу доступу до об'єктів). Для кодування запитів методів об'єктів і супутніх даних у протоколі SOAP використовуються тексти мовою XML (Extensible Markup Language). Великою перевагою протоколу SOAP є його простота. Внаслідок своєї простоти цей протокол може бути легко реалізований на багатьох пристроях. Протокол SOAP (Simple Object Access Protocol) може працювати на верхньому рівні будь-якого стандартного протоколу. Але саме можливість його роботи на верхньому рівні таких стандартних Internet-протоколів, як протокол передачі гіпертекстових файлів HTTP (Hypertext Transfer Protocol) і протокол SMTP (Simple Mail Transfer Protocol – простий протокол пересилання пошти), дає змогу пакетам даних проходити через системи мережного захисту без яких-небудь проблем, зв'язаних з можливістю з'єднання.

Тенденції розвитку клієнт-серверних систем приводять до переносу функціональності клієнта в спеціалізоване програмне забезпечення, яке також перебуває по стороні сервера і завантажується на робочу станцію користувача при його під’єднанні до інформаційної системи. При цьому робоча станція користувача надає лише середовище для виконання клієнтського програмного забезпечення.

Отже, розробка механізмів завантаження і виконання клієнтського програмного забезпечення та його взаємодії з серверами є актуальною науковою та прикладною задачею.

Постановка задачі

Полягає у класифікації інтерфейсів клієнт-серверної взаємодії, яка дозволила б робити передачу по каналу зв’язку об’єктів.

Основні результати досліджень

Технологію клієнт – сервер можна описати таким алгоритмом:

- користувач міняє стан якогось об’єкта клієнтської програми, який насамперед формує і посилає запит до сервера, вірніше – до програми, яка обробляє запити;
- ця програма проводить відповідні маніпуляції з даними, що є на сервері, відповідно до запиту, міняє стан пов’язаного об’єкта і передає його клієнтській програмі;
- клієнтська програма отримує об’єкт, враховує новий стан цього об’єкта і чекає подальших дій користувача. Цикл повторюється до того часу, поки користувач не завершить роботу з сервером.

Стандартне програмне забезпечення, що реалізується технологією клієнт-сервер, має: масштабованість (ефективне використання нарощеного апаратного забезпечення), стійкість в роботі, захист від несанкціонованого доступу і потужність під час роботи з великими проектами в галузі баз даних.

Модель клієнт-серверної взаємодії визначається передусім розподілом обов'язків між клієнтською та серверною програмами. У нашому випадку серверна програма надаватиме інформацію або інші послуги програмам, які звертаються до неї, а клієнтська програма буде використовувати ці сервіси.

Спрощена схема реалізації взаємодії між клієнтом і сервером, показана на рис. 1.

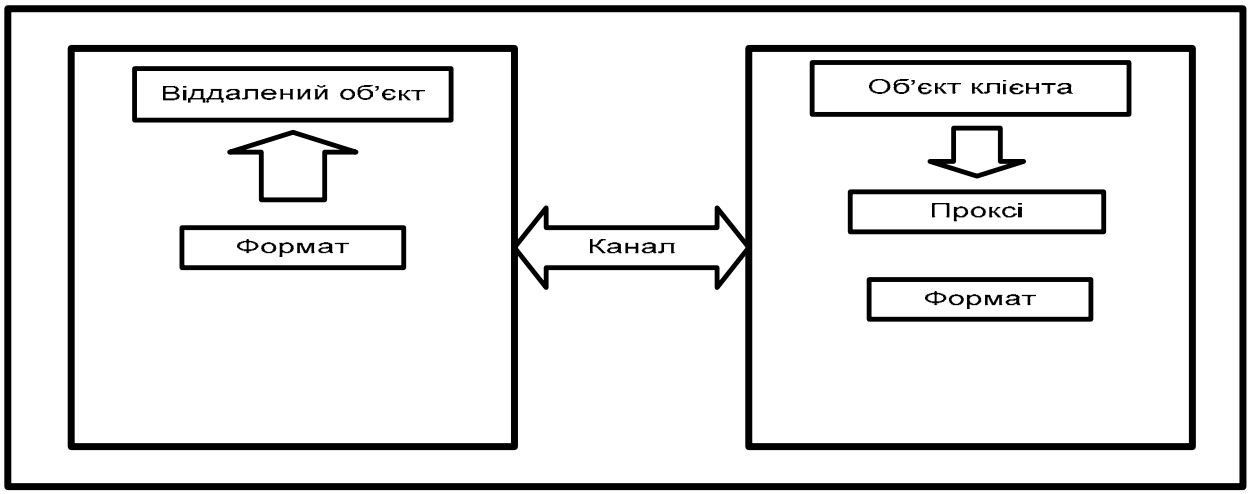

Рис. 1. Спрощена схема реалізації взаємодії між клієнтом і сервером

Класифікації засобів модульної взаємодії між клієнтом і сервером наведені в таблиці.

Класифікації засобів модульної взаємодії між клієнтом і сервером

Класифікації модульної взаємодії між клієнтом і сервером			
Активація режиму	Стан віддаленого об'єкта	Основа протоколів	Форматування
Сервер-активованих об'єктів	За значенням	HTTP-канал	XML формат
Клієнт-активованих об'єктів	За посиланням	TCP-канал	Двійковий формат

Класифікація засобів модульної взаємодії між клієнтом і сервером відбувається так [9–10], як наведено в таблиці.

1. Активація режиму:

1.1. Сервер-активованих об'єктів: об'єкти існують у просторі з адресою сервера, клієнт отримує доступ до цих об'єктів за посиланням через проксі-сервер (рис. 1). Для того, щоб клієнт отримав доступ до об’єкта, необхідно активувати клієнт. Існує два режим для активації клієнта: один – виклик (Single Call), інший – тон (Single Ton).Один виклик відбувається кожен раз, коли запит клієнта об'єкт, новий екземпляр створюється для конкретного запиту. Всі запити клієнт буде послідовно і в тому ж порядку будуть обслуговуватися екземпляри об'єкта, навіть якщо сервером ще не був виконаний попередній екземпляр. Один тон: сервер використовує тільки один екземпляр у будь-який момент часу. Якщо екземпляр не існує, сервер створює екземпляр, щоб обслуговувати всі подальші запити клієнта.

1.2. Клієнт-активованих об'єктів-це віддалені об'єкти, які активуються за запитом клієнта. При активації клієнта, коли клієнт намагається створити екземпляр віддаленого об'єкта, двосторонній зв'язок відбувається між клієнтом і сервером. Коли клієнт запитує віддалений об'єкт, віддалений додаток отримує повідомлення про активацію. Потім сервер створює екземпляр запитуваного класу і повертає посилання на об'єкт, щоб клієнтський додаток зміг його використовувати. На стороні клієнта за це відповідає проксі-сервер.

2. Стан віддаленого об'єкта (Remote Object). Застосовується для того, щоб використати об'єкт клієнта для доступу до функціональності об'єкта сервера через процеси. При цьому об'єкт сервера повинен бути віддаленими. Є два типи віддалених об'єктів:

2.1. За значенням. Використовується тоді, коли об'єкт сервера, отриманий у вигляді значення, тоді сервер використовує серіалізацію. Далі система сервера кодує об'єкт в послідовність бітів і передає його клієнту, який декодує цю послідовність бітів, а потім викликає методи.

2.2. За посиланням. У цьому типі сервера об'єктів, система сервера відсилає посилання на об'єкт клієнту. Клієнт-об'єкт потім використовує це посилання для виклику методів об'єкта сервера.

3. Основа протоколів. Надає канали, по яких повідомлення можуть обмінюватися віддалені об'єкти між собою. Для віддалених об'єктів через домени додатків. Передача об'єкта сервера на одному кінці і клієнтом об'єкта на інший. Об'єкти сервера вказує протокол і номер порту для отримання запиту. Клієнт об'єкта відсилає запит на сервер об'єкта за допомогою зазначеного протоколу та номера порту. Використовуються при цьому два канали протоколів:

3.1. HTTP-канал. Використовуйте HTTP-канал, якщо ви хочете спілкуватися через Інтернет, оскільки брандмауери не перешкоджають HTTP-зв'язку.

3.2. TCP-канал. Використовувати TCP-канал потрібно, якщо ви хочете спілкуватися всередині мережі. Якщо ви використовуєте TCP-канал, ви можете відкрити певні порти в брандмауерах між клієнтом і сервером.

4. Форматування. Необхідно перетворити дані у відповідному форматі, перш ніж передати їх по каналу. Форматування даних є об'єктами, які кодують і серіалізують дані у відповідному форматі (рис. 1). Форматування має бути реалізоване у інтерфейсі до передачі даних. Види форматування класів:

4.1. XML формат. Цей клас на основі SOAP. Цей протокол є модульним і не залежить від будь-якого конкретного транспортного протоколу, наприклад, HTTP або TCP. SOAP є ідеальним протоколом для зв'язку між додатками, які використовують несумісні архітектури. Тому що це XML-протокол.

4.2. Двійковий формат. Цей клас використовує виконавчі повідомлення для передачі інформації.

Двійковий формат простий і ефективніший ніж формат SOAP. Однак тільки додатки можуть прочитати двійковий формат.

Протокол Remote Object – це найшвидший спосіб зв'язку для передачі даних. Цей протокол використовує безпосередньо отримані дані. Remote Object як компонент використовує HTTP для передавання бінарних даних по каналу зв'язку. На рис. 2 показана архітектура Remote Object.

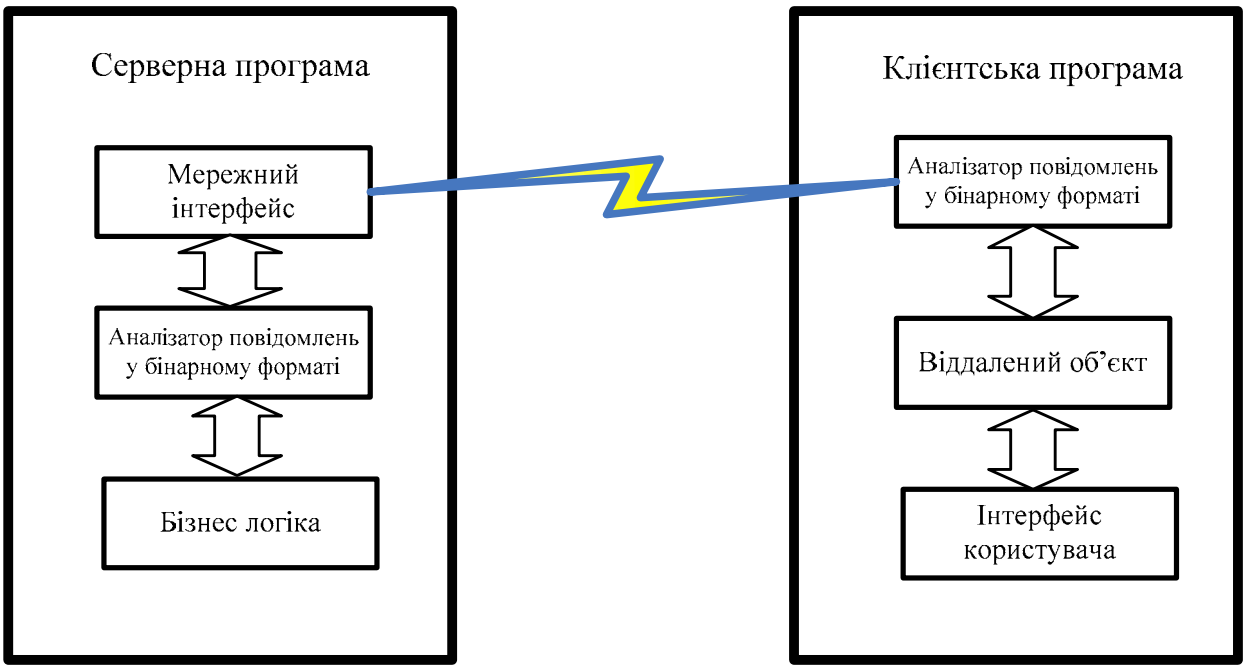

Рис. 2. Remote Object запит / відповідь моделі

Перевагами використання Remote Object є те, що значно швидший засіб зв’язку, ніж будь-які інші засоби зв'язку, другою перевагою використання Remote Object є те, що ми можемо надсилати та отримувати суворо типізовані об’єкти із сервера. При реалізації Remote Object використовується серіалізація та десеріалізація, яка відбувається на стороні сервера, який отримує можливість через мережний інтерфейс відсилати прості та складні об'єкти на сервер. Стани віддалених об'єктів можуть бути прив'язані до серверного об'єкта, шляхом додавання метаданих, тобто деяких тегів до класу об'єкта. Налаштування комунікацій згідно з протоколом Remote Object трохи складніше, ніж створення зв'язку клієнта з http-сервісом або web-сервісом. Протокол Remote Object у використанні з мережним інтерфейсом є більш ефективнішим, тому що він створює менше навантаження даними в двійковому форматі в результаті меншого розміру пакета даних.

Нові технології, при яких клієнтське забезпечення виконується по стороні сервера, значно відрізняються від традиційних клієнт-серверних технологій тому вимагають нової класифікації.

Висновки

Клієнт-серверна технологія покликана розділити роботу інформаційної системи в комп’ютерній мережі між сервером, який забезпечує основну функціональність інформаційної системи, та клієнтом, який здійснює ввід та попередню обробку інформації та її передачу на сервер і кінцеву обробку та візуалізацію отриманих з сервера даних.

Вироблення методології вибору технології клієнт-серверної взаємодії на фоні зростання кількості цих технологій і розвитку комп’ютерних мереж тепер є особливо актуальним.

Вищенаведена класифікація клієнт-серверної взаємодії дозволяє наочно оцінити придатність конкретних методів взаємодії клієнтських і серверних програм залежно від існуючих в кожному конкретному випадку обмежень.

1. Chris Giametta “Pro Flex on Spring”, 2009. – p. 445. 2. Роберт Дж. Оберг “Технология COM + Основы и программирование = Understanding and Programming COM+: A Practical Guide to Windows 2000 First Edition”. – М.: Вильямс, 2000. – С. 480. 3. Липаев В.В. “Обеспечение качества программных средств. Методы и стандарты”. – М.: Синтег, 2001. – С. 246. 4. Макгрегор Дж., Сайкс Д. “Тестирование объектно-ориентированного программного обеспечения”. – К: Диасофт, 2002. – С. 432. 5. Тамре Л. Введение в тестирование программного обеспечения. – М.: Вильямс, 2003. – С. 368. 6. Татарчук М. І. “Корпоративні інформаційні системи: Навч. посібник”. – К., 2005. – С. 245. 7. Мухамедзянов Н. “Java. Server applications”. – М: СОЛОН, 2003. – С. 267. 8. Орфалі Роберт, Ден Харкі “JAVA and CORBA in client server applications”. 9. Дуглас Камер, Девід Л. Стівенс // Сети TCP/IP, Т. 3. Разработка приложений типа клиент/сервер”. – М.: Вильямс, 2002. – С. 592. 10. Фленов М. Є. “Web-сервер глазами хакера: Проблемы безопасности Web-серверов; Ошибки в сценариях на PHP, Perl, ASP; SQL-инъекции”, 2005. – С. 365.