УДК 004.4

В. Ю. Коцюбинський, к. т. н., доц.; Р. С. Головаха

ВДОСКОНАЛЕННЯ ПРОЦЕСУ ПЕРЕДАВАННЯ ПОВІДОМЛЕНЬ З ВИКОРИСТАННЯМ FIX/FAST ПРОТОКОЛУ У ФІНАНСОВИХ КЛІЄНТ-СЕРВЕРНИХ СИСТЕМАХ

Розглянуто проблему виникнення затримок при передаванні великих об'ємів інформації у сфері електронної торгівлі на фінансових ринках, досліджено сучасні методи організації фінансових даних, реалізовано та здійснено детальний аналіз підходів до вирішення описаної проблеми.

Ключові слова: *передавання даних, FIX-протокол, FAST-протокол.*

Вступ. Із бурхливим розвитком індустрії інформаційних технологій та поширенням мережі Інтернет великої популярності набула сфера електронної торгівлі на фінансових ринках. Завдяки сучасним засобам автоматизації цього процесу, трейдеру більше немає необхідності перебувати безпосередньо на біржі, адже зараз він може купувати той чи інший актив зі свого торгового терміналу, віддаючи накази через Інтернет.

Значне поширення електронної торгівлі на фінансових ринках дало вагомий поштовх для розвитку технологій у цій сфері.

Аналіз попередніх досліджень. У 1992 році був розроблений Financial Information eXchange (FIX)-протокол для організації структури фінансових даних, що передаються. FIX-протокол – це міжнародний стандарт для обміну фінансовою інформацією між учасниками біржових торгів у режимі реального часу. FIX-протокол підтримується більшістю найбільших банків та бірж світу, а також електронними системами для інтернет-трейдингу [1].

Використання FIX має ряд переваг, таких як:

– ідеальна структура даних, що передаються, робить FIX-повідомлення легкими для читання як програмним кодом, так і людиною. Повідомлення FIX складаються з набору полів, які відділяються один від одного спеціальним кодом;

– FIX-протокол дозволяє здійснити перевірку цілісності даних, що передаються, оскільки визначає довжину тіла повідомлення і контрольну суму усіх байтів;

– підтримка повернення втрачених даних. FIX-протокол визначає поле, що означає порядковий номер повідомлення. Якщо послідовність даних порушена, будь-яка із сторін може зробити запит щодо втрачених повідомлень;

– безпека обміну даних. FIX-протокол підтримує авторизацію обох сторін та шифрування даних за різними методами [2].

З плином часу об'єми торгівлі фінансовими активами значно зросли, що призвело до збільшення об'єму даних, що передаються, виникненню значних затримок під час передавання повідомлень і неспроможності клієнт-серверних систем працювати у реальному часі.

У 2004 році у Нью-Йорку на конференції компанії FPL (FIX Protocol, Ltd. володіє правами та підтримує специфікацію FIX-протоколу) обговорювалась проблема про виникнення затримок при передаванні повідомлень, викликана збільшенням об'ємів даних, що передаються у фінансових мережах. Формат класичних полів FIX-протоколу вважається досить об'ємним, і їх обробка є занадто накладною. Затримка у доставці даних призвела до порушення можливості ведення торгів у реальному часі, що в свою чергу призвело до неспроможності трейдерів до торгівлі взагалі. Таким чином, FIX-протокол підлягав оптимізації.

У 2006 році була випущена перша версія FAST-протоколу (FAST 1.0). FAST (FIX Adapted

for STreaming) протокол – це технологія, спрямована на оптимізацію представлення даних у мережі. Він використовується для забезпечення високої пропускної здатності та малої затримки при передаванні даних між фінансовими системами шляхом стиснення повідомлень різними засобами [3 – 4].

FAST-протокол використовує різні технології для стиснення даних:

– кодування даних на рівні повідомлень, а саме використання шаблонів. Шаблони визначають дані, що передаються, та їх послідовність, що дозволяє виключити передачу тегів, які ідентифікують поля.

– кодування на рівні полів FIX-повідомлення. FAST-протокол поділяє поля на різні типи, що дозволяє або взагалі уникнути, або забезпечити часткове передавання відповідних значень.

– обмеження об'єму по типу даних, що передаються.

Використання Stop Bit. Кодування символу SOH, що розділяє поля FIX-повідомлення, у значення самих закодованих даних [5].

FAST-протокол дозволяє забезпечити значне стиснення даних при їх передаванні, але виникає інша проблема – це додаткові процеси при транспортуванні повідомлення, а саме кодування та декодування. При неефективній реалізації функціональності цих процесів, час на їх виконання буде досить великим, тому застосування FAST може взагалі втратити будь-який сенс.

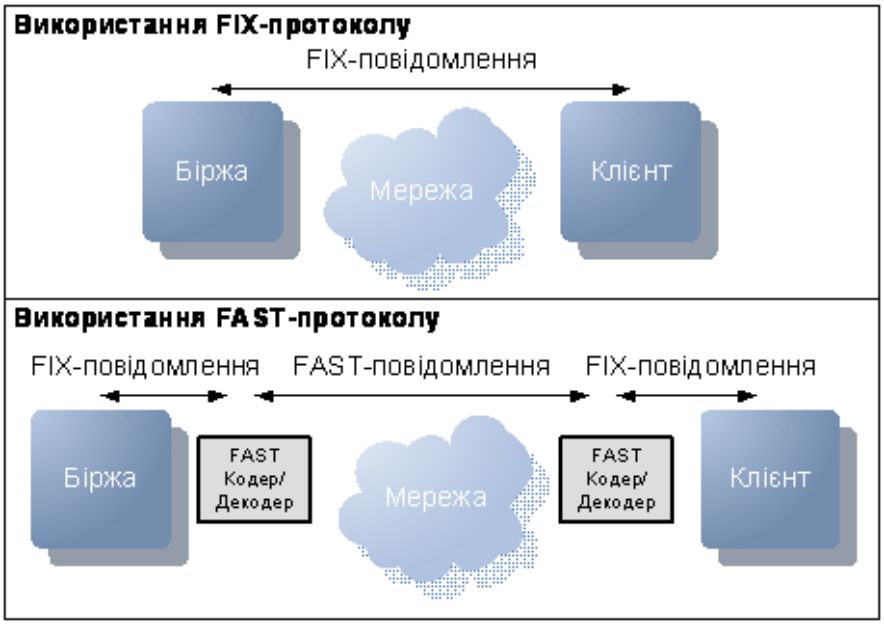

Рис. 1. Порівняння використання протоколів FIX та FAST при транспортуванні повідомлень.

Сучасні електронні системи для інтернет-трейдингу забезпечують створення торгових операцій через графічний інтерфейс, тому усі фінансові дані представляються трейдеру у чистому вигляді. На програмному рівні це означає, що після процесу декодування, система обробляє FIX-повідомлення і передає відповідні значення на відображення їх користувачу.

Таким чином, повна така система, яка забезпечує передачу даних від біржового сервера до їх графічного подання клієнту, вимагає наявності декількох процесів, що включають в себе: зчитування повідомлень з інформаційного біржового каналу, декодування даних, обробку повідомлень, передавання відповідних значень клієнту.

Так як електронна торгівля на фінансових ринках відбувається у реальному часі, затримка при передачі даних має бути достатньо малою, таким чином актуальність мінімізації часу на виконання описаних вище процесів є очевидною.

Метою роботи є підвищення пропускної здатності системи при передаванні фінансових

даних від біржового сервера до отримання відповідних значень для клієнта.

Матеріали та результати досліджень. Зараз існують відомі способи оптимізації часу на виконання розглянутих вище процесів. Проведемо детальний аналіз і реалізуємо програмно кожний із методів.

1. Розпаралелювання потоків. Тобто реалізація програмного коду з можливістю виконання різних задач у різних потоках. Цей підхід забезпечує асинхронну роботу декількох процесів одночасно, що в свою чергу може підвищити швидкість виконання коду у число разів, що залежить від кількості створених потоків.

З метою збільшення швидкодії виконання задач, доцільно реалізовувати програмний код таким чином щоб одночасно працювало не більше потоків, ніж число ядер процесорів у системі. В іншому випадку задачі будуть виконуватись по черзі, що відповідає синхронному виконанню, тому це втратить будь-який сенс використання паралельних потоків.

Так для системи, що розробляється, доцільно використовувати чотири потоки, що будуть здійснювати таку функціональність як:

– вичитування закодованих даних (FAST-повідомлення) з інформаційного каналу;
– декодування повідомлень у формат FIX;
– обробка FIX-повідомлень;
– передавання відповідних даних одному або декільком клієнтам.

Очевидно, що для розглянутої системи використання підходу розпаралелення потоків може підвищити швидкість програми, тобто зменшити затримку при передаванні даних від біржового сервера до 4-ох разів (для 4-ох ядерного процесора).

2. Фабрика об'єктів. Для програм, головним призначенням яких є транспортування даних, створення нових об'єктів є критичним моментом, так як сприяє виділенню пам'яті, що в свою чергу є трудомістким процесом, який уповільнює усю систему. Тому для уникнення зайвих затримок доцільно створювати усі необхідні об'єкти при ініціалізації системи, що на практиці не завжди є можливим.

Так для розглянутої системи кожний раз при надходженні нових даних від біржового сервера необхідно створювати новий об'єкт FIX-повідомлення. Цей об'єкт буде зберігати декодовані дані FAST-повідомлення для їх подальшої обробки. Фабрика об'єктів дозволяє частково позбутися періодичності створення нових об'єктів. Реалізація цієї функціональності виконується таким чином:

– при ініціалізації системи створюється визначена кількість FIX-об'єктів, що будуть використані по мірі необхідності;
– після використання усіх об'єктів, що були створені під час ініціалізації, при подальших запитах до фабрики кожного разу будуть створюватися нові об'єкти.

Очевидно, що у будь-якому випадку цей підхід має переваги, тобто впливає на зменшення часу при транспортуванні даних до клієнта. Але дана величина прямо пропорційна кількості повідомлень, що передаються, від біржового сервера та розміру, відповідно, самої фабрики об'єктів, тобто кількості об'єктів, що будуть створені при ініціалізації системи.

Проведемо тестування фабрики, запросивши 1 млн. нових FIX-об'єктів:

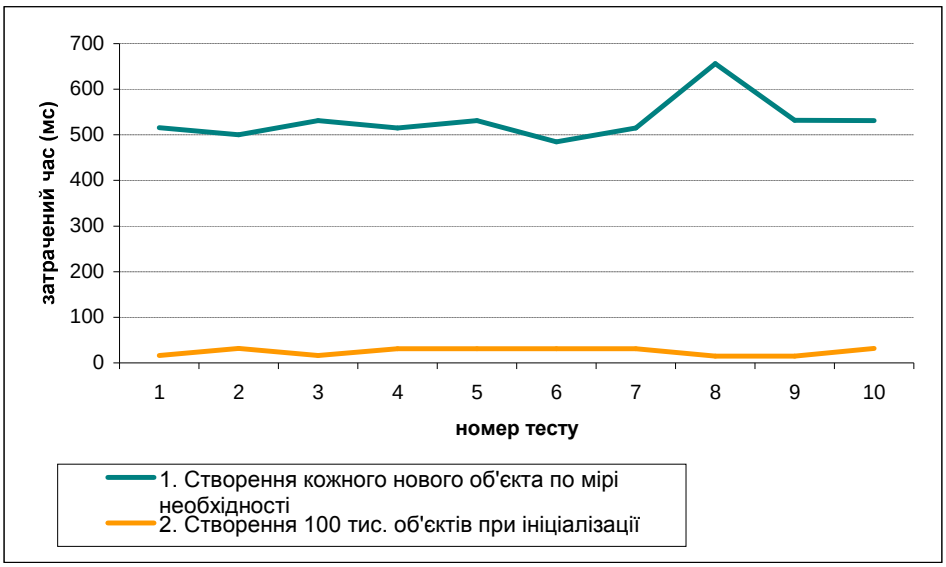

Рис. 2. Аналіз застосування фабрики об'єктів

Результати тесту на рис. 2 показують, що час, витрачений на створення нового об'єкту кожного разу за необхідністю, набагато більший (приблизно у 20 разів) у порівнянні з часом, що наближається до нуля при використанні фабрики об'єктів.

Ми можемо визначити будь-який розмір фабрики, тобто кількість об'єктів, що будуть створюватись при ініціалізації системи, що впливатиме лише на оперативну пам'ять.

3. Робота безпосередньо з масивами байтів та застосування побітових операцій. Оскільки вся цифрова інформація у комп'ютерних системах представляється у вигляді байтів, очевидно, що робота безпосередньо з ними вимагає найменших затрат часу у порівнянні з приведенням байтів до тих чи інших типів даних, які є більш зручними для роботи.

Так як перед нами стоїть задача підвищення пропускної здатності системи передавання даних, тобто зменшення часу на обробку повідомлень, доцільно, по можливості, забезпечити роботу програмного коду безпосередньо з байтами. Застосуємо цей підхід при декодуванні вхідних даних, тобто FAST-повідомлень, які надходять з біржового сервера у вигляді масивів байтів:

1)
```
offset=0;
while (offset < buffer.length-1) {
   value = (value << 7) | (buffer[offset++]);
}
value = (value << 7) | (buffer[offset++] & Byte.MAX_VALUE);
```

2)
```
Integer intDecodedVal = Integer.parseInt(strBuffer);
StringBuilder intDecBinSb = new StringBuilder(Integer.toBinaryString(intDecodedVal));
for(int i = intDecBinSb.length()-8; i >=0; i-=8) {
   intDecBinSb.deleteCharAt(i);
}
int value = Integer.parseInt(intDecBinSb.toString(), 2);
```

Робота безпосередньо з масивами байтів у прикладі 1) дозволяє застосувати побітові операції, які разом дають значні переваги при використанні (приблизно у 10 разів у порівнянні з прикладом 2)). Оскільки очевидно, що програмний код прикладу 2) включає в себе зайві операції, такі як приведення байтів у двійковий код, що є зайвим у порівнянні з прикладом 1) та створення нових об'єктів, що було описано у пункті 2.

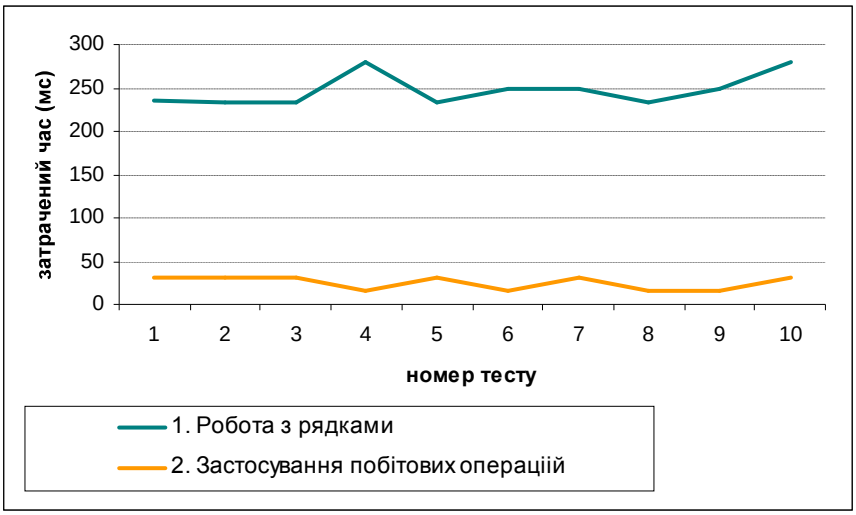

Рис. 3. Аналіз застосування побітових операцій

4. Позбавлення універсальності програмного коду. Очевидно, що універсальність коду в значній мірі зменшує його об'єм, але у той самий час і зменшує його швидкодію. Для більшості програм, цей момент не має критичного значення, тому йому не приділяється особлива увага. Але для програмних систем транспортування повідомлень, де потік даних є надзвичайно високим і головним критерієм оцінювання є швидкість передавання, універсальність коду може значно вплинути на час обробки інформації.

Рис. 4. Аналіз застосування універсальності коду

У системі, що розробляється, для досягнення найефективнішого результату необхідно забезпечити розділення усієї універсальної функціональності. Реалізуємо цей підхід для процесу декодування повідомлень, так як FAST-протокол забезпечує розподілення всіх даних на категорії, що визначаються наступним чином:

1) Fields operators. FAST-протокол описує 7 типів операторів полів, що визнають послідовність дій, які мають застосовуватися до того чи іншого поля FIX-повідомлення;
2) Nullable fields. Для таких полів декодоване значення "0" для числових типів даних або "" для рядкових означає передачу значення "null", що має різне застосування у відповідності до інших параметрів поля;

3) Data types. FAST-протокол описує 6 типів даних, що вимагають відповідні правила, які застосовуються для їх декодування.

Таким чином, розділивши цю функціональність, отримаємо $7 \cdot 2 \cdot 6 = 84$ класи, що визначає досить велику об'ємність, але в той же час значний виграш у часі (приблизно у 20 разів, що показано на рис. 4).

Ці дослідження відображають результати вдосконалення існуючих підходів для реалізації процесу FAST-кодування / декодування на замовлення компанії jettekfix.com.

Висновки. У цій роботі розглянуто проблеми, що є актуальними при передаванні великих об'ємів даних у системах електронної торгівлі на фінансових ринках, оптимізовано сучасні підходи розв'язання цих проблем та проведено аналіз їх використання. У результаті виконання цього дослідження доведено, що використання представлених прийомів дозволяє підвищити пропускну здатність системи передавання даних з біржового сервера приблизно у 4 рази, та, зокрема, процесу декодування FAST-повідомлень у 50 разів.

СПИСОК ЛІТЕРАТУРИ

1. FIX protocol [Електронний ресурс] // Режим доступу: http://ru.wikipedia.org/wiki/Financial_Information_eXchange.

2. Lamoureux Robert. Financial information exchange protocol / Robert Lamoureux // FIX Protocol Ltd. – 2001. – 280 p.

3. FAST protocol [Електронний ресурс] // Режим доступу: http://en.wikipedia.org/wiki/FAST_protocol.

4. Rosenborg David. FAST Protocol Technical Overview. / David Rosenborg. London: FIX Protocol Ltd. – 2006. – 10 p.

5. Rosenborg David. FAST Specification / David Rosenborg. London: FIX Protocol Ltd. – 2006. – 45 p.

Коцюбинський Володимир Юрійович – к. т. н., доцент кафедри автоматики та інформаційно-вимірювальної техніки.

Головаха Роман Сергійович – студент кафедри автоматики та інформаційно-вимірювальної техніки.

Вінницький національний технічний університет.