

СЕКЦІЯ XIII. КОМП'ЮТЕРНА ТА ПРОГРАМНА ІНЖЕНЕРІЯ

DOI 10.62731/mcnd-03.10.2025.002

ВЗАЄМОДІЯ З БІТКОЇН-МЕРЕЖЕЮ З ПРОГРАМНОГО СЕРЕДОВИЩА COMMON LISP

Жикін Юрій Сергійович

аспірант, асистент

КПІ ім. Ігоря Сікорського, Україна

Онай Микола Володимирович

канд.техн.наук, доцент

КПІ ім. Ігоря Сікорського, Україна

Вступ

Біткоїн [1] – це перша система електронних грошей, яка не має центрального органу контролю та центрального емітента. Завдяки своїм унікальним властивостям система швидко набула популярності і протягом 17 років з моменту появи вже стала важливою складовою світової фінансової інфраструктури. Основними властивостями Біткоїна є глобальна доступність (для здійснення Біткоїн-платежів достатньо лише доступу до Інтернету), стійкість до цензури (відсутність центрального органу, який може приймати рішення щодо легальності тієї чи іншої транзакції) і строга обмеженість пропозиції (автоматизований алгоритм емісії гарантує абсолютну дефляційність Біткоїна, що робить його цікавим інструментом для довготермінових інвестицій). З інженерної точки зору, Біткоїн-система – це однорангова комп'ютерна мережа, що складається з вузлів, які верифікують та розповсюджують непідтверджені транзакції, а також агрегують їх у ланцюг блоків – реєстр підтверджених транзакцій.

Common Lisp (CL) [2] – це стандартизований мультипарадигмовий діалект мови програмування Lisp, який підтримує поширені імперативний, функційний та об'єктно-орієнтований стиль програмування, а також надає потужні інструменти метапрограмування,

основним з яких є система макрофункцій, яка дозволяє користувачам створювати нові синтаксичні конструкції для конкретної предметної області за допомогою функцій, які виконуються під час компіляції програми і приймають оперують безпосередньо програмним кодом. Одним з найбільш важливих унікальних інструментів мови CL є інтерфейс інтерактивної розробки REPL (англ. *Read-Eval-Print Loop* – цикл читання, виконання і виведення), який дозволяє створювати функції, виконувати їх і інспектувати їх результати навіть під час виконання програми. Завдяки REPL-інтерфейсу, гнучкій системі розширення мови і динамічній типізації програмне середовище CL є дуже зручним для швидкої розробки прототипів програмного забезпечення, і при цьому забезпечує високу швидкість виконання програм завдяки компіляції у машинний код.

У цій роботі описується розроблена бібліотека інтерфейсів для взаємодії з Біткоїн-мережею з програмного середовища CL. Основною підставою розробки бібліотеки є відсутність подібних інтерфейсів у відкритому доступі, що ускладнює інтеграцію доступу до Біткоїн-мережі у програмні системи, що використовують середовище CL, зокрема AllegroGraph [3]. Також оцінюється швидкодія розробленого інтерфейсу десеріалізації Біткоїн-блоків і окреслюються плани подальшого вдосконалення розробленої бібліотеки.

Структура бібліотеки та порівняння швидкодії

Модулі CL називаються пакетами. Набори модулів (пакетів) CL називаються системами. Розроблена система складається з 4-х пакетів:

- ***bp.core*** – об'єктна модель блоків та транзакцій, допоміжні структури даних, спрощена реалізація правил консенсусу, а також високорівневі інтерфейси для взаємодії з мережею, зокрема основний інтерфейс системи – *bp.core:chain-supplier*;

- ***bp.crypto*** – криптографічні інструменти, зокрема оголошення зовнішніх функцій для виклику процедур з криптографічної бібліотеки *libsecp256k1*, яка реалізує криптографічні підписи ECDSA та Шнора на основі еліптичної кривої SECP256K1 [4];

- ***bp.net*** – рівень однорангової мережі, що реалізує підмножину типів мережеских повідомлень, а також надає примітивну реалізацію інтерфейсу *bp.core:chain-supplier*, яка отримує блоки безпосередньо з мережі, але не підтримує отримання транзакцій за ідентифікатором;

- ***bp.rpc*** – RPC-клієнт для вузла однорангової мережі Bitcoin Core [5], який реалізує повний перелік методів RPC-інтерфейсу, а також надає

єдину на даний момент повноцінну реалізацію інтерфейсу *bp.core:chain-supplier*.

Основним інтерфейсом у розробленій системі є інтерфейс *bp.core:chain-supplier*, який дозволяє отримувати сутності об'єктної моделі ланцюга блоків за їхніми ідентифікаторами: ідентифікатора блока за його позицією у ланцюгу, а також самого блока та транзакції за їхніми ідентифікаторами. На рисунку 1 наведено UML-діаграму інтерфейсу *bp.core:chain-supplier*.

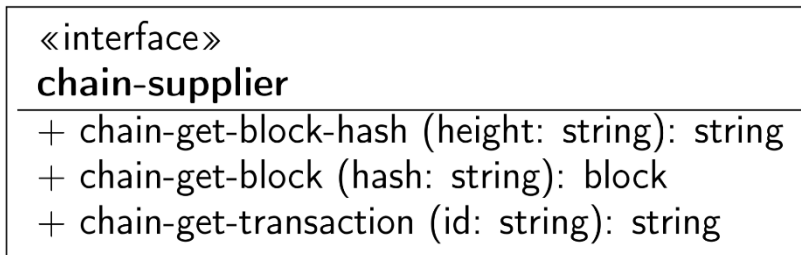


Рис. 1. **Діаграма інтерфейсу *bp.core:chain-supplier***

Приклад використання реалізації інтерфейсу *bp.core:chain-supplier* на основі RPC для отримання транзакції за ідентифікатором:

```

CL-USER> (bp:with-chain-supplier (bp.rpc:node-rpc-connection
                                :url "http://user:pass@localhost:8332")
          (bp:chain-get-transaction "0e3e2...12098"))
#<BP.CORE.TRANSACTION:TX 0e3e2...12098>
    
```

У таблиці нижче наведено порівняння швидкості читання і десеріалізації Біткоїн-блоків за допомогою інтерфейсу розробленої бібліотеки та за допомогою інтерфейсу популярної бібліотеки для взаємодії з Біткоїн-мережею з середовища Python.

Операція, 100 повторень	Час виконання, с	
	bp	python-bitcoinlib
Десеріалізація блока	7.501	10.184
Читання блока через JSON-RPC	49.906	37.013

Кожна операція (десеріалізація блока та читання блока через JSON-RPC) була повторена для однакових наборів з 100 блоків. Як видно з таблиці, швидкість десеріалізації блока за допомогою розробленого інтерфейсу є вищою, ніж за допомогою інтерфейсу бібліотеки Python, що

є очікуваним при порівнянні швидкодії компільованого та інтерпретованого програмного коду. Втім, читання блока через JSON-RPC є повільнішим. Оскільки операція читання блока складається з обміну HTTP-повідомленнями і операції десеріалізації, ймовірним поясненням цього є порівняно низька швидкість HTTP-комунікації HTTP-паketу, що використовується у розробленій бібліотеці.

План подальшого вдосконалення розробленої бібліотеки:

- заміна HTTP-паketу на паket, який забезпечить більшу швидкість HTTP-комунікації;
- завершення реалізації підтримки розширення протоколу Taproot;
- розробка класу, що реалізує інтерфейс *bp.core:chain-supplier* шляхом прямого доступу до файлів локального сховища вузла Біткоїн-мережі, з метою уникнення RPC-взаємодії за можливості.

Висновки

В цій роботі представлена бібліотека для взаємодії з Біткоїн-мережею з програмного середовища Common Lisp, основною підставою розробки якої була можливість інтеграції з програмним інтерфейсом графової бази даних AllegroGraph. Бібліотека надає інтерфейс для отримання з мережі сутностей об'єктної моделі ланцюга блоків Біткоїна. Швидкість десеріалізації блоків є вищою за швидкість аналогічної операції популярної бібліотеки для програмного середовища Python. Програмний код розробленої бібліотеки розміщено у відкритому доступі з ліцензією MIT на платформі колаборативної розробки програмного забезпечення GitHub, а також внесено в реєстри паketів Quicklisp та Ultralisp [6].

Список використаних джерел:

1. S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. URL: <https://bitcoin.org/bitcoin.pdf> (дата звернення: 27.09.2025).
2. Common Lisp HyperSpec. URL: <https://www.lispworks.com/documentation/HyperSpec/Front/index.htm> (дата звернення: 27.09.2025).
3. AllegroGraph Lisp Quick Start. URL: <https://franz.com/agraph/support/documentation/lisp-quickstart.html> (дата звернення: 27.09.2025).
4. libsecp256k1. URL: <https://github.com/bitcoin-core/secp256k1> (дата звернення: 27.09.2025).
5. Bitcoin Core integration/staging tree. URL: <https://github.com/bitcoin/bitcoin> (дата звернення: 27.09.2025).
6. BP – Bitcoin Protocol (in Lisp). URL: <https://github.com/rodentrabies/bp> (дата звернення: 27.09.2025).