A drawing of a building

AI-generated content may be incorrect.

**МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ  
 НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
імені ІГОРЯ СІКОРСЬКОГО»**

**Навчально-науковий інститут прикладного системного аналізу  
Кафедра системного проектування**

**Звіт**

**про виконання практичної роботи №4  
з дисципліни «Паралельні обчислення»**

Виконав:  
студент III курсу, групи ДА-21  
Сімков Микита Дмитрович

Прийняв:

асистент Яременко В. С.

Київ – 2025

**ПОСТАНОВКА ЗАДАЧІ**

1. Розробити клієнт-серверний застосунок для вирішення завдання з лабораторної роботи номер 1, передавши масив даних з клієнта на сервер, а потім – отримавши результат назад на сторону клієнта. Для виконання основного завдання дозволено використовувати лише платформні (WinSock, POSIX) та вбудовані засоби роботи з сокетами.
2. Самостійно розробити протокол прикладного рівня (application protocol) для взаємодії клієнта з сервером. Для цього врахувати декілька кроків в процесі взаємодії:
3. надсилання даних та конфігурації обчислень (наприклад, вказати кількість потоків для виконання обчислень)
4. надсилання команди (та отримання відповіді на команду) для початку обчислень
5. надсилання команди для запиту статусу і результату обчислень.
6. Додати до розробленого серверу підтримку підключення декількох клієнтів одночасно. Додатковий бал можна отримати у випадку наявності двох клієнтів, один з яких буде розроблений на мові, відмінній від мови серверу та першого клієнту (другий клієнт дозволено створювати з використанням скриптових мов).
7. Застосунок повинен коректно оброблювати виняткові ситуації як на стороні клієнту, так і на стороні серверу і адекватно реагувати на них. Без завершення своєї роботи. Обов’язковим є коректна обробка порядку кодування байтів у повідомленні.
8. В протоколі роботи необхідно навести опис розробленого застосунку. До цього опису повинні входити обґрунтування вибору протоколу передачі даних, а також архітектурний опис клієнта.
9. Занести до протоколу роботи опис розробленого протоколу прикладного рівня у вигляді таблиці, що включає: перелік всіх команд, аргументи команд та їх опис, список можливих відповідей на команду.
10. Занести до протоколу роботи UML діаграму викликів взаємодії серверу та клієнту, починаючи від запуску клієнту, до завершення роботи.
11. Надати висновок, що повинен містити аналіз та опис проблем з котрими зіштовхнувся студент, або з якими може зіштовхнутися розробник при організації міжпроцесової взаємодії.

**ХІД РОБОТИ**

Було створено 2 програми. Клієнт та сервер, які контактують 1 з одним по заздалегіть заданому протоколу. Повідомлення відправляються в форматі TLV. Якщо з довжиною та даними все зрозуміло, то тип передає команду, що потрібно виконати.

Сервер

* Приймає дані від клієнта.
* Обробляє матрицю у багатьох потоках відповідно до конфігурації.
* Повертає оброблені дані клієнту.

Протокол прикладного рівня (Application Protocol) визначає як будуть оброблятись повідомлення, що надходять до застосунку. У таблиці нижче наведено опис команд та можливих відповідей згідно протоколу TLV. Команди реалізовано у вигляді першого 1 байту.

|  |  |  |  |
| --- | --- | --- | --- |
| Код команди (TLV тип) | Довжина (Length) | Призначення | Очікувана відповідь від сервера |
| 0x01 | 4 байти | Клієнт надсилає кількість потоків (int32, big-endian). | - |
| 0x02 | 8 байт + N байт (матриця) | Клієнт надсилає матрицю  int32 rows, cols, далі матриця (int32 у big-endian). | - |
| 0x03 | 0 байт | Команда для запуску багатопотокової обробки | - |
| 0x05 | 0 байт | Запит на перевірку поточного стану обробки. | TLV-відповідь 0x06, значення — 2 байти: is\_matrix\_loaded, is\_processing. |
| 0x04 | - (використовується як відповідь, не команда) | Сервер надсилає клієнту оброблену матрицю. | - |
| 0хFF | Довжина повідомлення | Сервер відправляє повідомлення про помилку | - |

UML-діаграма роботи нашого застосунку.

A diagram of a diagram

AI-generated content may be incorrect.

Рисунок 1 — UML-діаграма застосунку.

Тепер спробуємо запустити декілька клієнтів для перевірки що все працює. Через реалізацію клієнти можуть підключатись поки не вичерпаютсья ресурси сервера.

A black and white image of a computer

AI-generated content may be incorrect.

Рисунок 2 — Запуск серверу.

A screenshot of a computer program

AI-generated content may be incorrect.

Рисунок 3 — Робота з декількома клієнтами.

A black screen with white text

AI-generated content may be incorrect.

Рисунок 4 — Лог клієнта.

Тепер спробуємо запустити клієнт написаний іншою мовою.

Для легкості напишу клієнт мовою python та бібліотекою struct, аналогічно буду користуватись побітовим кодуванням.

A screen shot of a computer code

AI-generated content may be incorrect.

Рисунок 5 – виконання клієнту на пайтоні

Також зроблю перевірку роботи повідомлень про помилку спеціально написавши клієнт з помилкою в повідомленнях

A black background with white text

AI-generated content may be incorrect.

Рисунок 6 – вивід помилки для 2 клієнту

**ВИСНОВКИ**

У результаті виконання лабораторної роботи було розроблено клієнт-серверний застосунок для обробки матриць у багатьох потоках. Клієнт надсилає матрицю на сервер разом із кількісю потоків, які будуть використовуватися для обчислень. Сервер обробляє дані, розподіляючи завдання між потоками, і повертає результати клієнту. Для забезпечення одночасного підключення декількох клієнтів сервер створює для кожного клієнта свій потік.

Протягом розробки було створено протокол прикладного рівня для взаємодії між клієнтом і сервером, який включає команди для передачі даних, запуску обчислень, перевірки статусу та отримання результатів. Передачу даних зроблено за допомогою бібліотеки socket та команд bind, listen, send. Також було додано обробку виняткових ситуацій, таких як розрив з'єднання або некоректні дані, що дозволяє програмі працювати стабільно.