Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Проміжне програмне забезпечення мультикомп'ютерної системи з топологією "кільце" двонаправленого зв'язку
Назва теми

КвРКІ 101058.21.01.08 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Виконав: студент III курсу, група КІ2с-21-1 _____ О.І. Моїк
Підпис, Ініціали, прізвище

Керівник _____ Д.М. Медзатий
Підпис, дата, Ініціали, прізвище

Нормоконтролер _____ І.О. Засорнова
Підпис, дата, Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Т.О. Говорущенко
Підпис, Ініціали, прізвище

«14» червня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень Бакалавр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О. Говорущенко

"10" 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Моїку Олександру Ігоровичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Проміжне програмне забезпечення мультикомп'ютерної системи з топологією "кільце" двонаправленого зв'язку

Керівник проекту (роботи) Медзатий Дмитро Миколайович, д.т.н., доцет.

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 15.02.2024 р. № 8

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2024 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити

Теоретичне дослідження проблеми та шляхи вирішення

Проєктування системи для топології «кільце» та вибір програмно-апаратного забезпечення для реалізації

Програмна реалізація мультикомп'ютерної системи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Схема типів топологій для мультикомп'ютерних систем

Результати тестування передачі повідомлень між пристроями

Діаграма класів

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийн
Нормоконтроль	Засорнова І.О., доцент кафедри КІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІС		

. Дата видачі завдання «_10_»_____01_____2024 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Приміт
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2024	виконан
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2024	виконан
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2024	виконан
4	Робота над розділом 2 – вибір компонентів для проєктування мультикомп'ютерної системи	01.04.2024	виконан
5	Робота над розділом 3 – проєктування мультикомп'ютерної системи типу подвійне кільце	29.04.2024	виконан
6	Оформлення пояснювальної записки згідно вимог	25.05.2024	виконан
7	Попередній захист ВКР	30.05.2024	виконан
8	Захист ВКР на засіданні ЕК	Червень 2024 року	

Студент _____ О.І. Моїк
Підпис Ініціали, прізвище

Керівник роботи _____ Д.М. Медзатий
Підпис Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 101058.21.01.08 ПЗ	Пояснювальна записка	63		
			Графічні матеріали			
2		КвРКІ 101058.21.01.08 Е8	Схема типів топологій для мультикомп'ютерних систем	1		
3		КвРКІ 101058.21.01.08 Е8	Результати тестування передачі повідомлень між пристроями	1		
4		КвРКІ 101058.21.01.08 Е8	Діаграма класів	1		

КвРКІ 101058.21.01.08 ВП

Зм	Арк	№ докум	Підпис	Дата
Розробив		Мох		24.06.24
Перевір.		Медзатий		24.06
Н. контр.		Засорнова		
Затв.		Говорущенко		24.06

Відомість проекту

Літера	Аркуш	Аркушів
У	1	1

ХНУ, КІ2с-21-1

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Проміжне програмне забезпечення мультикомп'ютерної системи з топологією “кільце” двонаправленого зв'язку».

Автор роботи: Моїк Олександр Ігорович.

Керівник роботи: Медзатий Дмитро Миколайович.

Пояснювальна записка: 63 с., 40 рис., 0 табл., 4 дод., 62 джерела.

Графічна частина: 3 креслення.

МУЛЬТИКОМП'ЮТЕРНА СИСТЕМА, ТОПОЛОГІЯ КІЛЬЦЕ, ПОДВІЙНЕ КІЛЬЦЕ, ПРОГРАМУВАННЯ С++, ТОПОЛОГІЯ МЕРЕЖІ

Метою дипломної роботи є розробка та аналіз проміжного програмного забезпечення для мультикомп'ютерних систем з двонаправленою топологією "кільце", а також оцінка ефективності та оптимізація даної системи для підвищення її продуктивності і відмовостійкості. Об'єктом дослідження є мультикомп'ютерні системи з двонаправленою кільцевою топологією.

Предметом дослідження є програмне забезпечення, що взаємодіє з апаратними компонентами таких систем для оптимізації обробки даних та забезпечення мережевих комунікацій. Під час проведення даного дослідження було використано метод комплексного аналізу існуючих рішень та методи моделювання для вивчення і аналізу ефективності та можливостей оптимізації розглядуваних систем.

Підпис студента

10.06.2024

Дата

ЗМІСТ

ВСТУП

Сучасний світ цифрових технологій вимагає від науковців та інженерів створення все більш складних та ефективних систем обробки даних. Однією з ключових концепцій у розвитку обчислювальних мереж є мультикомп’ютерні системи, що забезпечують високу продуктивність та надійність завдяки розподіленій обробці даних. Особливе місце серед таких систем займає мультикомп’ютерна система з двонаправленою топологією «кільце», яка відрізняється здатністю до самовідновлення та балансування навантаження.

Мультикомп'ютерна система – це комплекс, що складається з численних взаємопов'язаних комп'ютерів, які працюють як єдине ціле. У такій системі кожен комп'ютер зазвичай має власну пам'ять та процесор, але вони можуть обмінюватися даними та завданнями через мережу.

Топологія «кільце» дозволяє кожному вузлу системи спілкуватися безпосередньо з двома сусідніми вузлами, створюючи замкнутий цикл. Це забезпечує високу стійкість системи до збоїв, адже у випадку виходу з ладу одного з вузлів, передача даних може бути швидко перенаправлена через інший напрямок. При цьому, двонаправлене зв'язування значно збільшує загальну пропускну спроможність та ефективність комунікацій.

Ще однією перевагою мультикомп'ютерних систем є їх здатність забезпечувати високу надійність. Якщо один з комп'ютерів у системі відмовить, інші можуть продовжувати роботу, забезпечуючи безперервну функціональність системи.

В рамках цієї дипломної роботи буде розроблено та аналізовано проміжне програмне забезпечення для мультикомп’ютерної системи з такою топологією. Програмне забезпечення, як інтерфейс між апаратними та програмними компонентами системи, відіграє ключову роль у забезпеченні її стабільності та продуктивності. Ця робота спрямована на створення моделі програмного

забезпечення, що ефективно управлятиме ресурсами системи, оптимізуючи передачу даних та використання апаратних ресурсів.

Метою роботи є моделювання та проектування моделі програмного забезпечення, що зможе забезпечити високу продуктивність та надійність мультикомп'ютерної системи з двонаправленою топологією «кільце». Результати дослідження мають практичне значення для розвитку мультикомп'ютерних систем, використання їх у різних сферах діяльності та підвищення їхньої ефективності у вирішенні складних обчислювальних та аналітичних завдань.

1 ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖУВАНОЇ ПРОБЛЕМИ

1.1 Аналіз предметної області і виявлення наявних проблем і завдань

Мультикомп'ютерна система представляє собою комплекс обчислювальних пристроїв, які працюють у взаємодії та координації для вирішення обчислювальних завдань. У відміну від традиційних багатопроцесорних систем, де кілька процесорів об'єднані в одному корпусі, мультикомп'ютерні системи складаються з окремих вузлів, які можуть знаходитися на різних фізичних пристроях і спілкуватися через мережу передачі даних [1].

Основна архітектура мультикомп'ютерних систем базується на розподіленій архітектурі, де кожен вузол системи може мати власну пам'ять, процесор та інші обчислювальні ресурси. Вузли взаємодіють між собою за допомогою мережі передачі даних, що може бути реалізована як локальна мережа (LAN), глобальна мережа (WAN) або спеціалізована інтерконектована мережа [2, 3].

Для забезпечення ефективної взаємодії між вузлами мультикомп'ютерних систем застосовуються різноманітні технології передачі даних та обміну повідомленнями. Це може бути реалізовано за допомогою різних протоколів комунікації, таких як Message Passing Interface (MPI), Remote Procedure Call (RPC), або через високо рівневі бібліотеки та середовища розробки, які спрощують програмування паралельних та розподілених застосунків.

Основними характеристиками мультикомп'ютерних систем є масштабованість, гнучкість та висока продуктивність. Їхня масштабованість дозволяє легко збільшувати кількість вузлів для розв'язання більш складних завдань. Гнучкість полягає у здатності до розгортання на різних апаратних платформах та використанні різних технологій зв'язку. Висока продуктивність досягається завдяки паралельному виконанню обчислень на різних вузлах, що дозволяє значно зменшити час виконання складних завдань.

Мультикомп'ютерні системи забезпечують потужний інструмент для вирішення великих обчислювальних задач у наукових, інженерних, технічних та

комерційних галузях, і вони продовжують займати важливе місце в сучасних обчислювальних середовищах.

Основою для підвищення ефективності обчислень слугують передові технології та здатність системи до паралельної обробки великого обсягу задач.

Відповідно до класифікації Флінна [4, 5], паралельні обчислювальні машини поділяються на два основних типи: SIMD, де виконується одна інструкція над множиною даних, і MIMD, що дозволяє виконувати багато інструкцій одночасно. Це розмаїття підходів відкриває широкі можливості для оптимізації та адаптації обчислювальних систем під конкретні завдання.

Щодо архітектурних рішень, то вони можуть включати:

- конвеєрну та векторну обробку, яка передбачає послідовне виконання обчислювальних операцій у кілька етапів, з ефективною передачею результатів від одного етапу до наступного;
- системи SIMD, які складаються з багатьох однотипних обчислювальних елементів з власною пам'яттю, що працюють під управлінням єдиної програми;
- системи MIMD, що пропонують велику гнучкість завдяки можливості кожного обчислювального елемента виконувати індивідуальні програми, що особливо ефективно в мультипроцесорних системах зі спільною пам'яттю;
- багатопроцесорні системи з SIMD процесорами, що представляють собою складні багаторівневі архітектури, здатні одночасно використовувати векторні операції та масово-паралельні обчислення, надаючи користувачам широкий спектр можливостей для розв'язання складних задач.

Мультикомп’ютерна система – це складна структура, що включає в себе різноманітні компоненти для ефективної обробки і обміну даними [6, 7].

Мультикомп’ютерну систему можна розділити на такі ключові категорії:

1. Персональні комп'ютери (ПК) – ці пристрої зазвичай служать одній особі для виконання особистих чи професійних завдань і відрізняються портативністю та персоналізацією. Вони включають у себе ноутбуки, настільні ПК, планшети та інші подібні гаджети [8].

2. Спеціалізовані комп'ютерні системи – це категорія включає різні типи комп'ютерів, призначених для специфічних завдань або застосунків. Сюди входять сервери, що надають ресурси іншим машинам через мережу, робочі станції для важливих обчислювальних операцій, таких як 3D моделювання чи обробка відео, кластери для комплексних обчислень та вбудовані системи, інтегровані у багатофункціональні пристрої [9, 10].

3. Провідні з'єднання – використання Ethernet у локальних мережах для створення стабільних провідних з'єднань між пристроями, включаючи комп'ютери та маршрутизатори. USB порти забезпечують універсальне підключення периферійних пристроїв, в той час як HDMI використовується для передачі аудіо та відео між електронними пристроями [11, 12].

4. Бездротові з'єднання – Wi-Fi стандартизує бездротові локальні мережі, дозволяючи безперебійне з'єднання між пристроями без використання кабелів, а Bluetooth сприяє зв'язку на короткі дистанції між гаджетами, такими як навушники та клавіатури. Мобільні мережі забезпечують доступ до Інтернету на переносних пристроях за допомогою технологій 4G та 5G [14].

5. Програмне забезпечення та операційні системи – операційні системи керують основними функціями комп'ютерів, включаючи управління ресурсами, введення/виведення даних та процесами. Диспетчер завдань координує розподіл ресурсів між задачами, а системи управління ресурсами оптимізують їх використання для забезпечення стабільної роботи [15, 16].

6. Розподілена обробка даних – віртуальні машини дозволяють емулювати різні обчислювальні архітектури в ізольованому середовищі, що забезпечує виконання програм на різноманітних платформах. Контейнери, такі як Docker, надають ізольоване середовище для запуску програм, ділячись системним ядром з основною операційною системою [17, 18].

7. Розподілена операційна система – це тип операційної системи, який координує та управляє розподіленими ресурсами через мережу, спрощуючи

управління розподіленими програмами та сервісами. Вона забезпечує синхронізацію та ефективний обмін даними між різними вузлами системи.

Також необхідно звернути увагу на компоненти безпеки, відстеження стану системи та резервне зберігання даних, які є ключовими для забезпечення стабільності та захищеності мультикомп'ютерних мереж. Інтеграція цих елементів сприяє формуванню продуктивної та надійної мультикомп'ютерної інфраструктури.

У контексті мультикомп'ютерних систем, значущість обговорення паралельного програмування та взаємодії процесів не може бути недооцінена. В умовах, коли задіяно кілька обчислювальних вузлів, оптимальне розподілення завдань та координація обміну даними між ними є вирішальними для ефективності системи.

Паралельне програмування дозволяє розбивати комплексні завдання на менші підзадачі, які можуть бути виконані одночасно на різних вузлах, що забезпечує значне прискорення процесу обробки даних. Цей підхід є особливо цінним в мультикомп'ютерних установках, де кожен вузол може спеціалізуватися на певній частині задачі.

Безпека в мультикомп'ютерних системах є пріоритетною, особливо через їх залежність від мережевих інтерфейсів для взаємодії. Такі заходи, як шифрування даних, процеси аутентифікації та механізми доступу, є критично важливими для захисту конфіденційності та цілісності інформації у таких системах [19].

На завершення, важливо відзначити постійний розвиток мультикомп'ютерних технологій, включаючи застосування хмарних рішень для гнучкого розширення обчислювальних можливостей та інтеграцію новітніх технологій, таких як квантові обчислення, які обіцяють трансформувати майбутнє цифрової обробки даних.

1.2 Сфери застосування мультикомп'ютерних систем

На сучасному етапі мультикомп'ютерні системи еволюціонують у напрямку формування динамічних мережних структур, які можуть розгортатися на різних масштабах, навіть до глобального рівня. Ці системи володіють змінною конфігурацією та асинхронним функціонуванням компонентів, спрямованим на виконання завдань зі слабко зв'язаними компонентами, такими як задачі перебору чи пошуку.

Мультикомп'ютерні системи надають прозорий та контрольований доступ користувачам до системи через Інтернет та забезпечують прозоре підключення не використовуваних обчислювальних ресурсів до мультикомп'ютера. Це дозволяє ефективно використовувати вільні ресурси для вирішення поточних завдань, забезпечуючи при цьому оптимальний рівень продуктивності та ефективності всієї системи.

Наукові дослідження різноманітні й вимагають високої обчислювальної потужності. Мультикомп'ютерні системи застосовуються для складних досліджень, таких як моделювання клімату, геноміка та астрофізика. У сфері інженерії, ці системи використовуються для комп'ютерного проектування, аналізу та моделювання в авіації, автомобілебудуванні та машинобудуванні. Фінансові ринки вимагають точних аналізів та прогнозів, і мультикомп'ютерні системи знаходять застосування у фінансовому аналізі, моделюванні ризиків та прогнозуванні цін.

Серед тенденцій розвитку варто виділити хмарні технології та квантові обчислення які стрімко набирають популярність через свою ефективність.

Хмарні технології (Cloud Technology) [20-22] – це технології надання комп'ютерних ресурсів, таких як обчислювальна потужність, зберігання даних та програмне забезпечення, онлайн-послуги через Інтернет, без необхідності купувати та обслуговувати власну інфраструктуру. Серед причин популярності хмарних технологій можна виділити такі пункти:

- підвищення безпеки – хмарні провайдери постійно вдосконалюють свої методи захисту даних, забезпечуючи високу ступінь безпеки для користувачів. Це включає шифрування даних, контроль доступу та регулярні аудити безпеки;
- інфрастуктура та обчислювальні потужності – із розвитком технологій та проблем, які вони вирішують необхідні великі обчислювальні можиливості, які самостійно важко підтримувати. Таким чином провайдери надають уже готові сервери для таких завдань у використання;
- гнучкість – окремо варто виділити гнучкість таких систем, оскільки потреби можуть змінюватися, через це хмарні технології передбачають високу гнучкість та маштабованість для задоволення потреб.

Рисунок 1.1 – Обчислювальна хмара [23]

Квантові обчислення мають потенціал значно збільшити продуктивність мультикомп'ютерних систем. Завдяки принципам квантової механіки, квантові комп'ютери можуть виконувати обчислення з неймовірною швидкістю та

ефективністю порівняно з класичними комп'ютерами. Це особливо важливо для складних задач, таких як моделювання молекулярних структур, оптимізаційні проблеми та великі обчислювальні моделі.

Мультикомп'ютерні системи, що складаються з великої кількості взаємопов'язаних комп'ютерів, вже широко використовуються для розробки та використання систем штучного інтелекту. Такі системи дозволяють паралельно обробляти величезні обсяги даних та виконувати складні алгоритми, що необхідні для навчання нейронних мереж та інших моделей ШІ.

Рисунок 1.2 – Квантові технології [24]

FDDI (Fiber Distributed Data Interface) може бути розглянутий як приклад мультикомп'ютерної системи, побудованої за топологією подвійного кільця (рисунок 1.3)

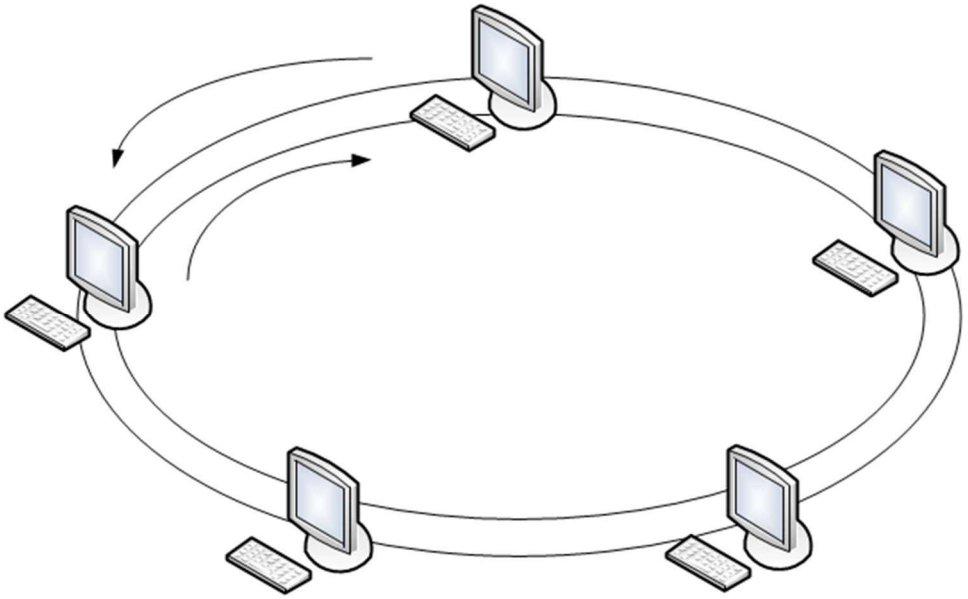


Рисунок 1.3 – Топологія подвійне кільце [25]

Технологія FDDI [26-29] була розроблена та стандартизована інститутом ANSI у 1988 році з метою поліпшення швидкості передавання даних та надійності локальних мереж. Це була перша технологія, яка використовувала волоконно-оптичні кабелі для передачі даних. Акронім FDDI означає "Fiber Distributed Data Interface" (розподілений оптоволоконний інтерфейс даних).

Основні характеристики технології FDDI [30]:

- швидкість передачі даних: 100 Мбіт/с;
- метод доступу до фізичного середовища: маркерне кільце;
- топологія: подвійне кільце;
- основне фізичне середовище: волоконно-оптичний кабель;
- максимальна довжина мережі: 200 км (2 х 100 км);
- максимальна кількість вузлів: 500.

Мережа FDDI складається з двох кілець на основі волоконно-оптичного кабелю, до яких підключені робочі станції. Одне кільце є основним, а інше - резервним. У звичайному режимі передачі даних використовується основне кільце. Резервне кільце використовується в разі обриву основного кабеля або виходу з ладу однієї з робочих станцій. Використання двох кілець підвищує надійність мережі

FDDI і забезпечує автоматичне відновлення її роботи за допомогою стандартних процедур.

1.3 Порівняльний аналіз переваг та недоліків існуючих рішень

Мультикомп'ютерні системи утворюють важливу складову цифрового ландшафту, надаючи можливість обробляти та обмінюватися масивами даних на небувалому рівні. Ці системи виконують роль складного оркестрового ансамблю, де кожен "музикант" - це процесор з унікальними можливостями, а "диригент" – це ретельно налаштована мережа, яка злагоджує роботу всієї системи. Розглядаючи мультикомп'ютерні системи з двонаправленою топологією "кільце", ми виявляємо унікальні переваги та виклики, які вони пропонують.

Переваги:

- збільшення продуктивності - системи дозволяють паралелізувати обробку даних, розділяючи завдання між різними вузлами;
- надійність - у випадку збою одного з вузлів, система зберігає свою працездатність завдяки існуванню альтернативних шляхів передачі даних у кільцевій топології;
- гнучкість масштабування системи легко розширюються за рахунок додавання нових вузлів, збільшуючи обчислювальну потужність без необхідності зміни загальної архітектури.

Оцінюючи цілісну картину мультикомп'ютерних систем, особливо тих, що побудовані на двонаправленій кільцевій топології, необхідно звернути увагу на їх здатність до самовідновлення та адаптації до змінних умов роботи. Система, що може динамічно реагувати на відмови та автоматично переконфігуруватися, забезпечуючи неперервність робочих процесів, є надзвичайно цінною в умовах, де високий рівень доступності даних є критичним. Водночас, складність управління такою системою та вимоги до безпеки вимагають розробки витончених рішень, які б забезпечували не тільки технічну ефективність, але й відповідали б високим стандартам захисту інформації.

Недоліки:

- складність управління – координація роботи численних вузлів та забезпечення їх ефективної взаємодії може бути складним завданням, особливо з урахуванням двонаправленої топології «кільце»;

- вимоги до безпеки з кожним доданим вузлом зростає кількість потенційних точок входу для зловмисників, що збільшує вразливість системи до кібератак;

- забезпечення консистентності даних у мультикомп'ютерних системах з двонаправленою топологією "кільце" забезпечення синхронізації даних між вузлами може бути складнішим через можливість циркуляції даних у обох напрямках.

Енергоефективність стає все більшою пріоритетною у мультикомп'ютерних системах. Застосування енергоефективних технологій та оптимізація використання ресурсів може значно знизити загальне споживання енергії.

Автоматизоване відновлення роботи систем також грає важливу роль. Системи можуть бути налаштовані на автоматичне перерозподілення завдань у випадку відмови одного або декількох вузлів. Це забезпечує високий рівень доступності ресурсів, навіть у випадку непередбачених ситуацій.

Інноваційність також є ключовою характеристикою сучасних мультикомп'ютерних систем. Топологія "кільце" відкриває можливості для інтеграції новітніх технологій та підходів, сприяючи постійному технологічному розвитку.

Розглядаючи ці аспекти, стає зрозуміло, що мультикомп'ютерні системи з топологією "кільце" пропонують унікальний набір можливостей та викликів, які потребують ретельного аналізу та уважного підходу при проектуванні та розгортанні. Системи захисту та аутентифікації: Забезпечення надійного рівня захисту є особливо важливим, адже мультикомп'ютерні системи можуть стати цілями для кібернападів. Реалізація надійних механізмів аутентифікації та оборони даних є критичною складовою їх безпеки.

1.4 Підходи до вирішення задачі за темою дослідження

У сучасному цифровому світі мультикомп'ютерні системи стають ключовими елементами обробки та розподілу великих масивів даних між різними вузлами. Перед тим, як поглиблюватися у специфіку архітектури та топології таких систем, необхідно визначити основні компоненти, які формують їхню основу.

Вузли мультикомп'ютерних систем, які можуть бути представлені у вигляді персональних комп'ютерів, спеціалізованих пристроїв або інших обчислювальних елементів, є фундаментом для обробки даних. Вони з'єднуються між собою за допомогою мережевих з'єднань, що можуть бути як провідними, так і бездротовими, в залежності від конкретних потреб та умов використання системи.

Серцевиною мультикомп'ютерних систем є програмне забезпечення, яке керує взаємодією між компонентами системи та забезпечує обмін даними. Це може включати в себе системи управління базами даних, операційні системи, програмне забезпечення для розподіленої обробки даних та інше.

Ключову роль у забезпеченні ефективної комунікації в мультикомп'ютерних системах відіграють протоколи зв'язку, такі як TCP/IP, UDP, HTTP, які встановлюють стандартизовані правила для передачі інформації між вузлами [31].

Ефективне функціонування мультикомп'ютерних систем неможливе без належної системи управління ресурсами, яка забезпечує раціональне використання пам'яті, обчислювальних потужностей та мережевих ресурсів.

Застосунки, що працюють на мультикомп'ютерних системах, виконують широкий спектр задач, адаптованих під конкретні потреби користувачів, починаючи від простих обчислень та закінчуючи складними аналітичними операціями.

Особливу увагу слід приділити архітектурі мультикомп'ютерних систем, яка може базуватися на розподіленій пам'яті, де кожен вузол має власну локальну пам'ять, що забезпечує високу швидкість доступу до даних, але вимагає координації для синхронізації даних між вузлами [32].

Вибір топології мережі має стратегічне значення, оскільки від нього залежить структура взаємодії пристроїв у системі. Кільцева топологія, зокрема, пропонує простоту та надійність завдяки послідовному з'єднанню вузлів у кільце, хоча й може стикатися з обмеженнями у швидкості обробки даних.

Врахування принципів енергоефективності та оптимізації ресурсів є важливим для підтримки сталого розвитку та зниження експлуатаційних витрат системи.

Адаптивність та масштабованість системи, які дозволяють легко нарощувати або скорочувати ресурси відповідно до змінних вимог, є ключовими для підтримки гнучкості та ефективності мультикомп'ютерних систем у динамічному інформаційному середовищі.

1.5 Постановка задачі

Цей розділ покликаний описати цілі та завдання розробки мультикомп'ютерної системи з топологією "кільце" двонаправленого типу, а також визначити ключові напрямки досліджень та розробок для її удосконалення.

Основною метою даної роботи є створення мультикомп'ютерної системи універсального призначення, яка базується на двонаправленій топології "кільце". В рамках роботи передбачається здійснення глибокого аналізу існуючих архітектурних рішень, з метою виявлення їх сильних сторін та обмежень при застосуванні в контексті розподілених систем з загальною пам'яттю.

В другому та третьому розділах роботи буде представлено всебічний аналіз проектування та реалізації системи, супроводжуваний обґрунтуванням вибору конкретних технічних рішень та їх впливу на загальну продуктивність та ефективність системи.

Під час розробки мультикомп'ютерної системи на основі двонаправленої топології "кільце" особлива увага буде приділена відбору та оптимізації ключових компонентів системи, включаючи обчислювальні вузли, комунікаційні інтерфейси

та системи зберігання даних. Ефективність кожного компонента критично важлива для забезпечення високопродуктивної роботи загальної системи.

Важливим аспектом розробки є досягнення оптимального балансу між вартістю та продуктивністю системи. Економічна ефективність рішення повинна бути обґрунтована через ретельний вибір компонентів з урахуванням їх ціни, продуктивності та сумісності.

На програмному рівні розробка відповідного програмного забезпечення є ключовою для забезпечення злагодженої роботи мультикомп'ютерної системи. Вибір програмного забезпечення повинен спрямовуватися на забезпечення ефективності, масштабованості та зручності управління ресурсами системи.

У підсумку, успішна реалізація проекту мультикомп'ютерної системи з двонаправленою топологією "кільце" вимагає комплексного підходу, що включає в себе виважений вибір апаратних та програмних компонентів, а також ефективне їх взаємодія для досягнення поставлених цілей.

1.6 Висновки

У вступному розділі дослідження ми розглянули основи функціонування обчислювальних систем, зокрема, ознайомились з класифікацією Флінна, що визначає чотири типи архітектур паралельних ЕОМ: від конвеєрної та векторної обробки до багатопроцесорних систем із SIMD процесорами. Така класифікація допомагає зрозуміти, як за допомогою паралельної обробки та використання швидкодійних елементів можна значно підвищити продуктивність обчислювальних систем.

Ми також розглянули ключові компоненти мультикомп'ютерних систем, де комп'ютери відповідають за обробку даних, мережеве з'єднання забезпечує взаємодію між вузлами, а програмне забезпечення керує загальним процесом і координує обмін даними. Важливу роль відіграють також протоколи комунікації, які встановлюють правила для передачі даних, та системи управління ресурсами, що оптимізують використання доступних потужностей.

Операційні системи, що є невід'ємною частиною мультикомп'ютерних систем, виконують функції посередника між користувачем та апаратним забезпеченням, спрощуючи управління різноманітними компонентами системи та забезпечуючи їх злагоджену взаємодію.

Топології мультикомп'ютерних систем відіграють ключову роль у формуванні структури підключення вузлів та організації комунікаційного простору, що впливає на ефективність обміну даними між компонентами системи.

Крім того, аналізуючи існуючі мультикомп'ютерні системи, ми змогли оцінити їхню ефективність, сфери застосування та потенційні можливості для подальшого розвитку та вдосконалення, враховуючи сучасні вимоги та тенденції в області обчислювальної техніки.

2 АРХІТЕКТУРА ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ

2.1 Поняття топології та її види

Топологія в мультикомп'ютерних системах визначає спосіб, яким комп'ютери та інші пристрої підключені між собою. Це може впливати на швидкодію, надійність та масштабованість системи. Ось кілька типових топологій, що використовуються в мультикомп'ютерних системах [33, 34]:

1. Зіркова топологія (рисунок 2.1). Базова топологія комп'ютерної мережі, де всі комп'ютери з'єднані з центральним вузлом, таким як комутатор, утворюючи фізичний сегмент мережі. Цей сегмент може функціонувати як самостійна мережа або як частина складної мережевої топології, зазвичай у формі дерева.

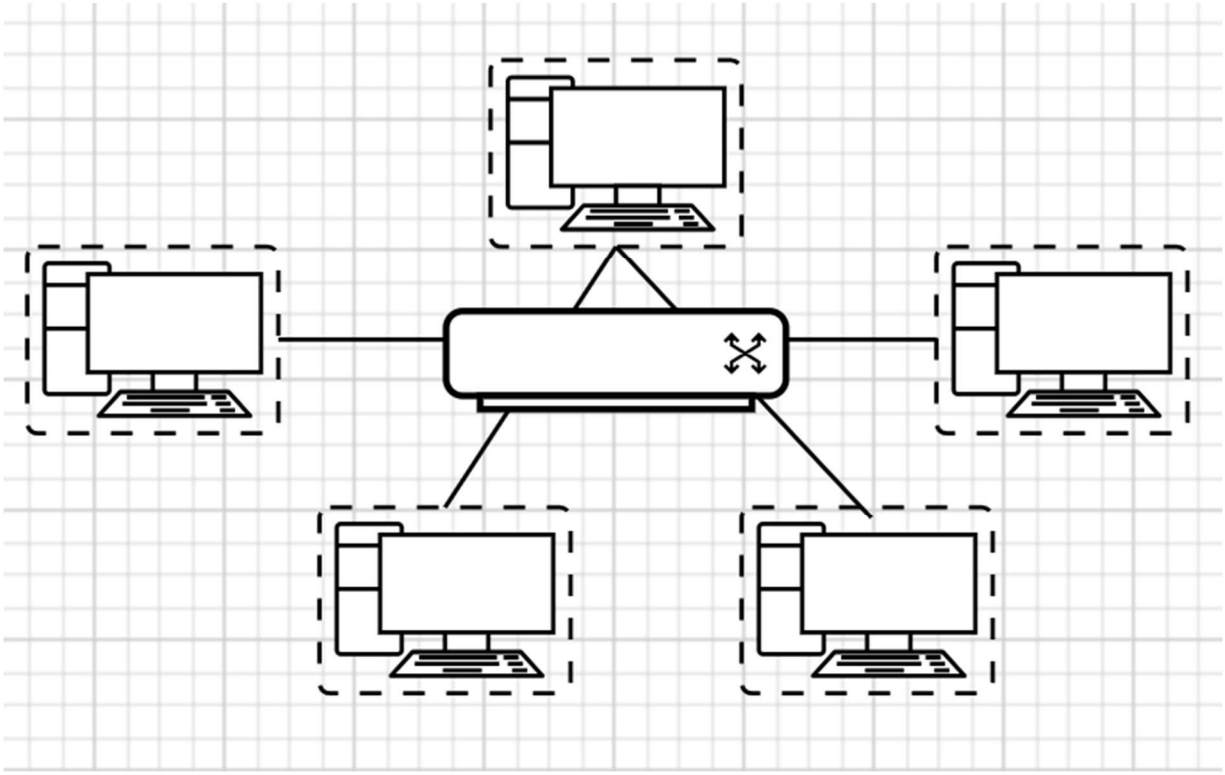


Рисунок 2.1 – Зіркова топологія

2. Топологія «шина» (рисунок 2.2) також відома як «загальна шина», передбачає однаковість мережного обладнання комп'ютерів та рівноправність всіх користувачів через її структуру. У такій мережі комп'ютери можуть передавати

дані лише послідовно через єдину лінію зв'язку. Це обмежує режим напівдуплексного обміну, де передача інформації відбувається в обидва напрямки, але не одночасно, а почергово. В іншому випадку інформація може бути спотворена через конфлікти або колізії.

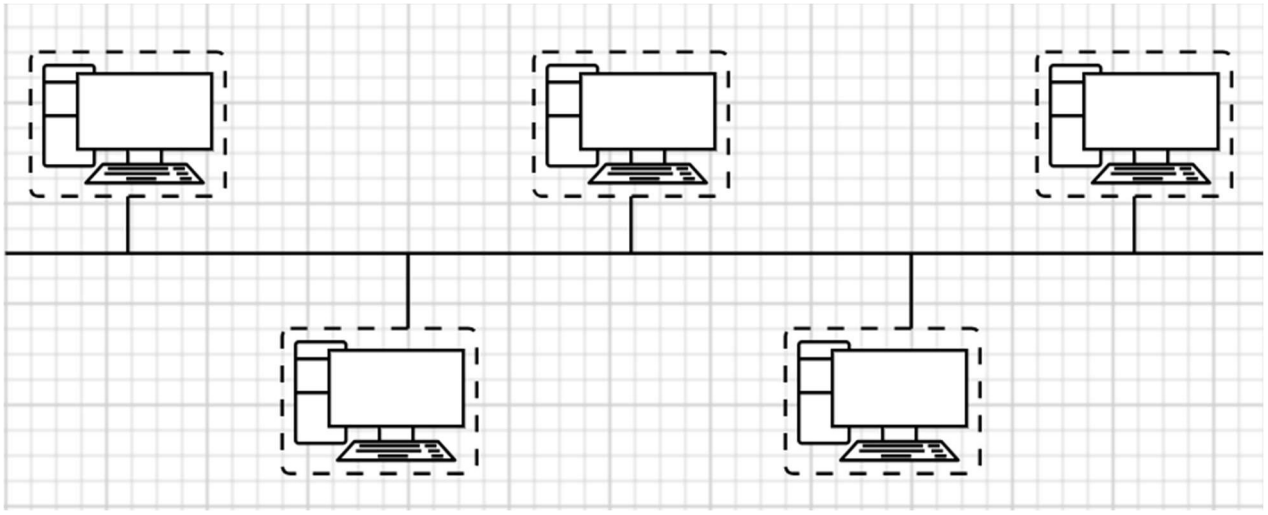


Рисунок 2.2 – Топологія шина

3. Коміркова топологія (рисунок 2.3), або топологія меш (mesh) виникає з повнозв’язної мережі шляхом видалення певних можливих з'єднань.

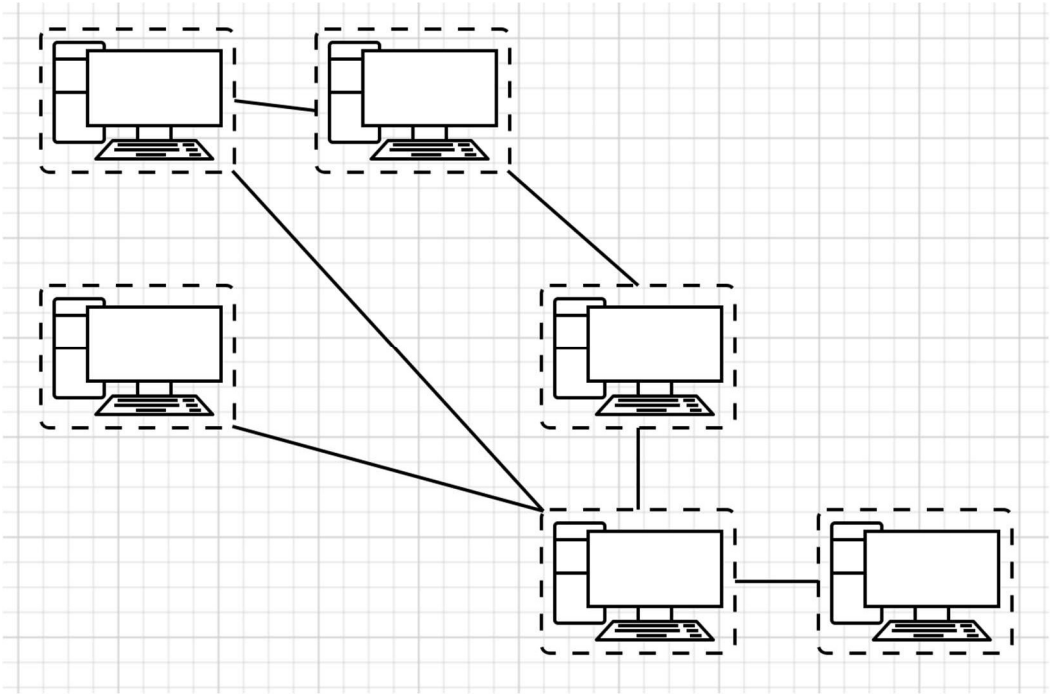


Рисунок 2.3 – Коміркова топологія

У мережі з такою топологією прямо з'єднуються лише ті вузли, які активно обмінюються даними, а для комунікації між вузлами, які не мають прямих зв'язків, використовуються транзитні маршрути через інші вузли. Мережа з меш-топологією може об'єднувати велику кількість вузлів і зазвичай характеризується глобальним масштабом.

4. Топологія дерево (рисунок 2.4) в мережній топології означає, що кожен вищий вузол має з'єднання з нижчими вузлами у вигляді зіркового маршруту, утворюючи комбінацію зірок. Це також відомо як ієрархічна зірка. Термін "дерево" походить з теорії графів. Перший вузол в такому дереві називається коренем, вищі вузли - батьківськими, а нижчі - дочірніми. Таким чином, кожен дочірній вузол, який має зв'язок з менш високими вузлами, стає для них батьківським.

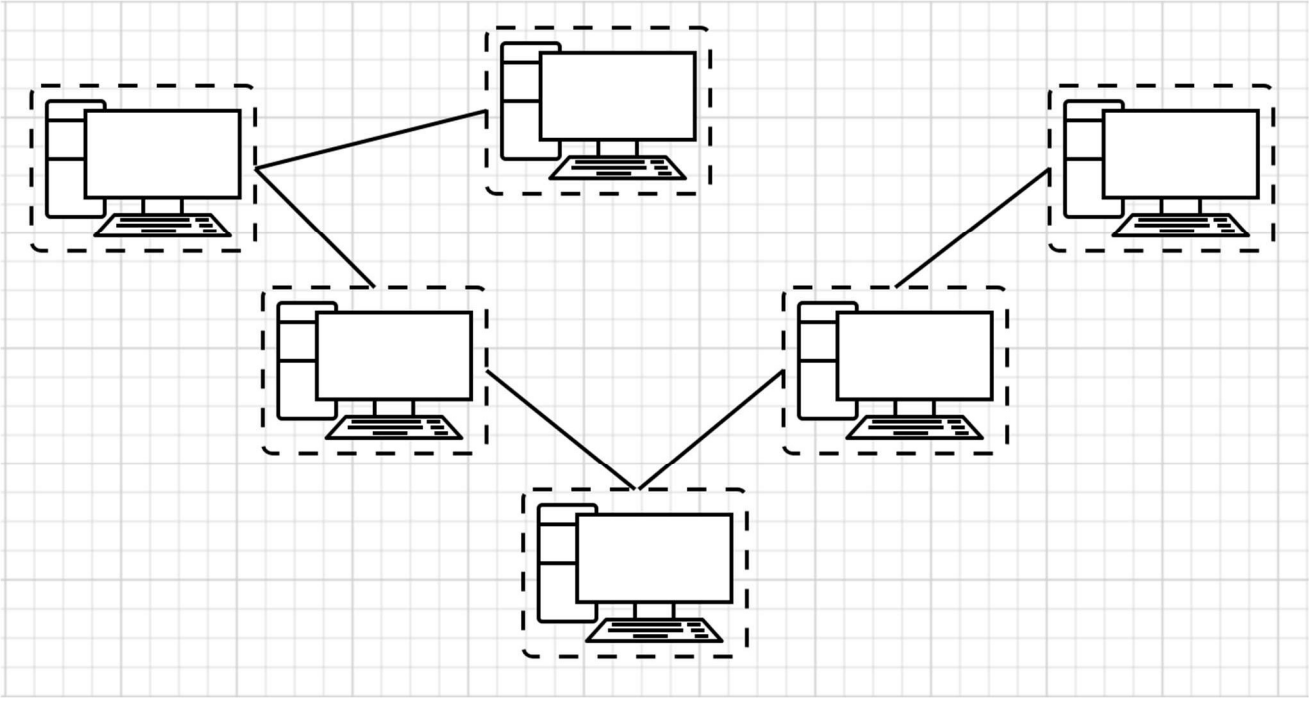


Рисунок 2.4 – Топологія дерево

5. Топологія кільце (рисунок 2.5). Кільце – це мережева топологія, в якій кожен комп'ютер має зв'язки з двома іншими комп'ютерами: він отримує інформацію від одного та передає іншому. На кожному з'єднанні працює лише один передавач і один приймач (це зв'язок точка-точка). Це дозволяє уникнути

необхідності використання зовнішніх термінаторів, як у випадку з топологією «зірка».

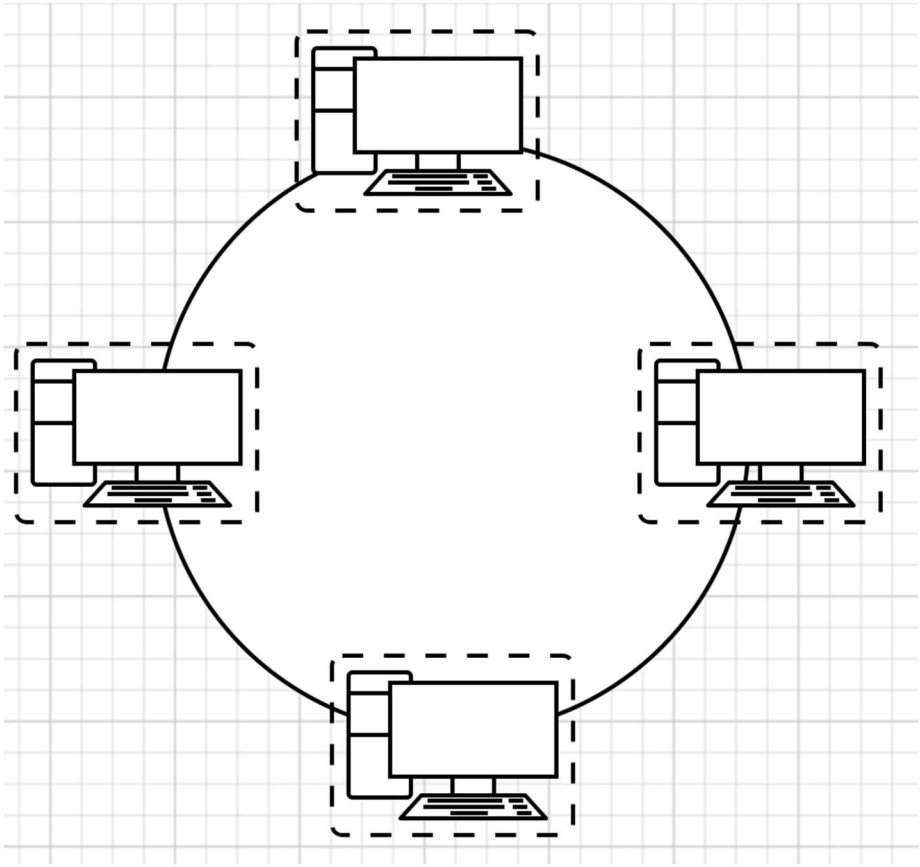


Рисунок 2.5 – Топологія кільце

2.2 Вибір топології «кільце» двонаправленого зв’язку

З серед трьох розглянутих топологій, «кільце» є найбільш вразливою до пошкоджень кабелю, тому в цій топології зазвичай передбачають прокладання двох (або більше) паралельних ліній зв'язку, одна з яких є резервною (рисунок 2.6).

Іноді мережу з топологією "кільце" будується на основі двох паралельних кільцевих ліній зв'язку, які передають інформацію у протилежних напрямках. Так і утворюється нова топологія “кільце” двонаправленого зв’язку.

Мета такого підходу - подвоїти швидкість передачі даних у мережі. При пошкодженні одного з кабелів мережа може переключитися на інший кабель, хоча максимальна швидкість передачі буде меншою [35].

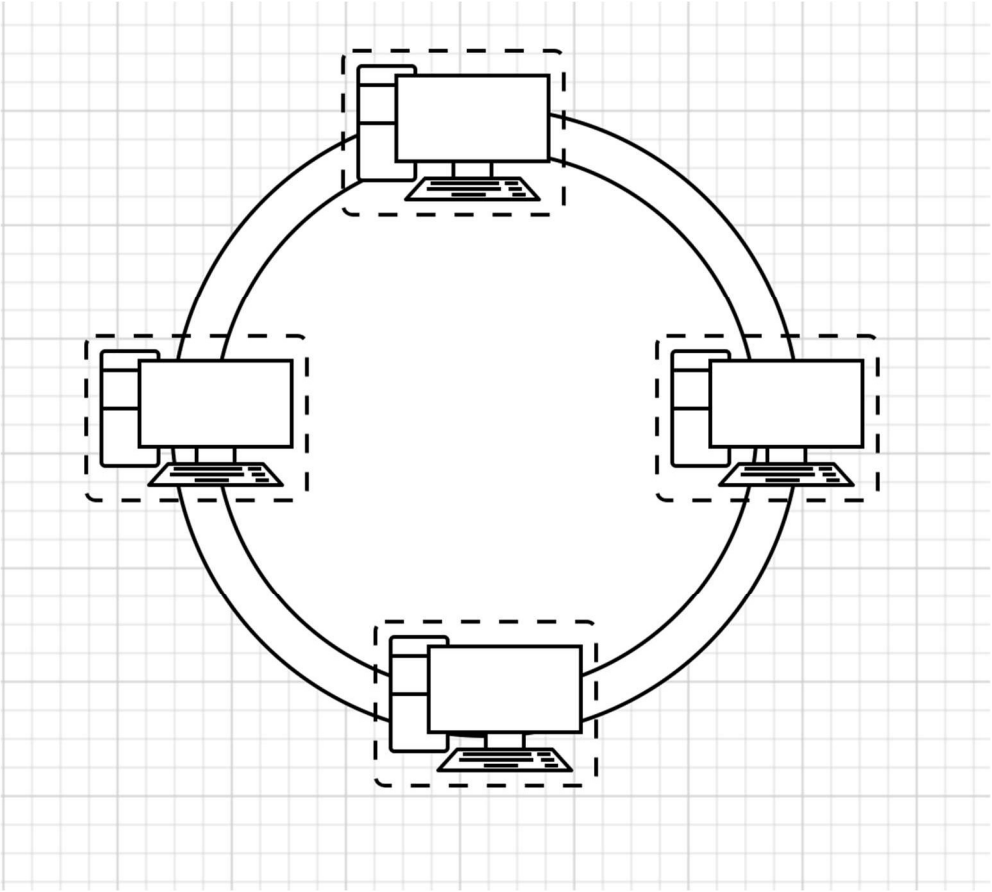


Рисунок 2.6 – Топологія подвійне кільце

Переваги:

- контроль над процесом доставки даних адресату;
- ефективне пересилання повідомлень, оскільки можна відправляти кілька повідомлень один за одним по кільцю;
- легкість відстеження вузлів, які працюють некоректно;
- протяжність мережі може бути значною, дозволяючи комп'ютерам підключатися один до одного на великих відстанях без використання спеціальних підсилювачів сигналу.

Недоліки:

- зі збільшенням кількості клієнтів швидкість роботи мережі сповільнюється, оскільки всю інформацію проходить кожен вузол;
- низька надійність мережі, оскільки відмова будь-якого вузла призводить до відмови всієї системи;

- для підключення нового клієнта необхідно відключити роботу мережі.

У кільцевій топології передача даних відбувається за допомогою технології «токену», яка пересувається вздовж кільця від одного пристрою до наступного. Коли пристрій бажає передати дані, він очікує, доки токен досягне його, після чого додає свої дані до токену і передає його далі по кільцю. Коли дані досягають призначеного пристрою, вони вилучаються з токену, і токен продовжує свій шлях по кільцю. Перший вузол з'єднаний з останнім вузлом, щоб утворити цикл [36].

2.3 Вибір апаратного забезпечення

Зважаючи на топологію «кільце» для побудови мультикомп'ютерної системи, яка включає 1 сервер та 3 клієнтів, необхідно ретельно підібрати апаратне забезпечення. Ось більш докладний опис необхідного обладнання:

1. Сервер – центральний вузол системи, який забезпечує управління мережею та надає ресурси для клієнтів. Для оптимальної продуктивності цей сервер повинен мати достатні ресурси (процесор, пам'ять, сховище) для обробки запитів від клієнтів та забезпечення надійності мережі.

2. Клієнтські комп'ютери – три клієнтські пристрої, які виконують різноманітні завдання і взаємодіють з сервером та іншими клієнтами. Вони можуть бути персональними комп'ютерами або робочими станціями з відповідними характеристиками, щоб виконувати потрібні завдання.

3. Кільцевий комутатор або концентратор – центральний пристрій, який створює фізичне кільце зв'язку. Це може бути комутатор Ethernet або концентратор Token Ring, здатний керувати передачею даних вздовж кільця [37].

4. Кабельна інфраструктура необхідно налагодити мережеву інфраструктуру для підключення всіх комп'ютерів до кільцевого комутатора. Це може включати в себе категорію кабелю, необхідну для підтримки швидкості передачі даних і довжини мережі.

5. Мережеві карти або інтерфейси - кожен комп'ютер має бути оснащений мережевою карткою або інтерфейсом, який дозволяє підключити його до кільцевої мережі [38].

6. Програмне забезпечення необхідно встановити необхідне програмне забезпечення на сервері та клієнтах для налагодження та керування мережею. Це може включати операційну систему, додатки для обміну даними та програми для моніторингу та керування мережею.

2.3.1 Сервер для мультикомп’ютерної системи

Один з прикладів сервера, який можна використати для побудови мультикомп'ютерної системи за топологією "кільце" з двонаправленим зв'язком, це Dell PowerEdge T340 (рисунок 2.7).

Рисунок 2.7 – Сервер Dell PowerEdge T340 [39]

Цей сервер є надійним та потужним рішенням для малого бізнесу або офісної мережі. Він оснащений процесором Intel Xeon E-2200 з можливістю розширення до 8 ядер, що забезпечує високу продуктивність обробки даних. Також сервер має до 64 ГБ оперативної пам'яті, що дозволяє обробляти великі обсяги даних та запитів розподілених на 4 слоти DIMM. Підтримка процесорів Intel Xeon E-2200 з можливістю розширення до 8 ядер. Підтримка до 8 жорстких дисків SATA або SAS або до 16 SSD SATA з можливістю розширення за допомогою контролера RAID. Слоти розширення PCIe для підключення додаткових компонентів, таких як мережеві адаптери або контролери RAID.

У даному сервері передбачені розширені можливості забезпечення безпеки та керування мережею. Він має вбудовані засоби моніторингу та діагностики, а також можливість встановлення додаткових програмних засобів для керування мережею.

Dell PowerEdge T340 також має достатньо розширених можливостей для підключення до кільцевої мережі, зокрема вбудовані мережеві інтерфейси та слоти розширення для додаткових мережевих адаптерів, якщо це необхідно.

Цей сервер також має ефективну систему охолодження та енергозбереження, що дозволяє економити електроенергію та забезпечує безперебійну роботу протягом тривалого періоду часу.

2.3.2 Комп’ютери для мультикомп’ютерної системи

Один з прикладів комп'ютера, який можна використати для побудови мультикомп'ютерної системи за топологією "кільце" з двонаправленим зв'язком, це HP ProDesk 600 G6 Microtower (рисунок 2.8).

Рисунок 2.8 – Комп’ютер HP ProDesk 600 G6 Microtower [40]

Цей комп'ютер відомий своєю надійністю та продуктивністю, що робить його ідеальним вибором для побудови мережевих систем. Деякі ключові характеристики включають:

- доступність широкого спектру процесорів Intel Core і Intel Xeon, які забезпечують високу продуктивність та ефективність обробки даних;
- підтримка до 128 ГБ оперативної пам'яті DDR4, що забезпечує достатній обсяг пам'яті для виконання різноманітних завдань;
- можливість встановлення жорстких дисків або SSD з різними обсягами пам'яті, що дозволяє зберігати великі обсяги інформації;
- вбудований гігабітний Ethernet-контролер для підключення до мережі з двонаправленим зв'язком;
- слоти розширення PCIe для підключення додаткових мережевих адаптерів або інших пристроїв;
- сумісність з різними операційними системами, такими як Windows, Linux або іншими;
- мікротауерний корпус забезпечує ефективне використання простору і дозволяє легко монтувати комп'ютери в стійки або на робочих столах.

2.3.3 Комутатори з підтримкою кільцевої топології

Cisco Catalyst 2960 Series Switch – це комутатор, який підтримує кільцеву топологію мережі (рисунок 2.9). Він відомий своєю надійністю та широкими можливостями для побудови мереж з різними топологіями, включаючи кільцеву. Цей комутатор має гігабітні швидкості передачі даних по всіх портах, розширені функції керування трафіком, можливості налаштування VLAN та QoS, а також вбудовані засоби захисту мережі, такі як ACL та портова безпека. Крім того, Cisco Catalyst 2960 Series Switch підтримує різні методи управління мережею, включаючи CLI та GUI. Цей комутатор є надійним та потужним рішенням для побудови мережі з кільцевою топологією і забезпечує високу продуктивність, надійність та безпеку мережі.

Рисунок 2.9 – Комутатор Cisco Catalyst 2960 Series Switch [41]

2.3.4 Маршрутизатори для забезпечення зв’язку між різними сегментами топології

Один з прикладів маршрутизатора, який можна використати для забезпечення зв'язку між різними сегментами кільцевої топології з двонаправленим зв'язком, це Cisco ISR 4000 Series Integrated Services Router (рисунок 2.10).

Цей маршрутизатор є надійним та потужним рішенням для побудови мережі з розподіленою топологією. Він має розширені можливості маршрутизації та комутації, які дозволяють ефективно керувати трафіком між різними сегментами мережі.

Рисунок 2.10 – Cisco ISR 4000 Series Integrated Services Router [42]

Cisco ISR 4000 Series може обробляти великий обсяг трафіку завдяки потужним процесорам та оптимізованим алгоритмам маршрутизації.

Цей маршрутизатор має різноманітні порти Ethernet, WAN та LAN, що дозволяє підключати його до різних сегментів мережі з двонаправленим зв'язком.

Cisco ISR 4000 Series має вбудовані засоби захисту мережі, такі як файрвол, VPN та IPS (Intrusion Prevention System), що забезпечує безпеку даних та конфіденційність інформації.

Маршрутизатор підтримує різні методи управління мережею, включаючи веб-інтерфейс, CLI та SNMP (Simple Network Management Protocol), що забезпечує зручність управління та моніторингу мережі.

Cisco ISR 4000 Series має вбудовані механізми забезпечення надійності, такі як резервне живлення, резервування інтерфейсів та механізми відновлення після збоїв, що забезпечує безперебійну роботу мережі.

2.3.5 Кабель для підключення до обох кілець

Для підключення мультикомп'ютерної системи за топологією "кільце" з двонаправленим зв'язком можна використовувати волоконно-оптичний кабель (оптичний кабель).

Волоконно-оптичний кабель складається з тонких скляних або пластикових волокон, через які передаються світлові сигнали. Цей тип кабелю ідеально підходить для кільцевих мереж з двонаправленим зв'язком через наступні переваги.

Волоконно-оптичні кабелі забезпечують значно вищу пропускну здатність порівняно з традиційними мідними кабелями, що дозволяє передавати великі обсяги даних на великі відстані. Сигнали у волоконно-оптичних кабелях не піддаються електромагнітній інтерференції, що робить їх ідеальним варіантом для використання в умовах, де важлива стійкість до зовнішніх впливів.

Також сигнали у волоконно-оптичних кабелях не випромінюються назовні, що робить їх менш вразливими до перехоплення даних зовнішніми атаками, забезпечуючи високий рівень конфіденційності та безпеки.

Волоконно-оптичні кабелі дозволяють передавати сигнали на значні відстані без значних втрат сигналу, що робить їх ідеальним варіантом для побудови кільцевих мереж.

Один з прикладів волоконно-оптичного кабелю, який можна використовувати для підключення мультикомп'ютерної системи за топологією "кільце" з двонаправленим зв'язком, це Multimode Duplex Fiber Optic Cable.

Цей тип кабелю має два волоконні шнури (duplex), що дозволяють передавати дані в обидва напрямки одночасно. Кожен шнур складається з багатьох тонких скляних або пластикових волокон, які вкладені в захисну оболонку. Волоконно-оптичні кабелі мають різний діаметр волокон та різну пропускну здатність.

Наприклад, Multimode Duplex Fiber Optic Cable (рисунок 2.11) може мати такі характеристики:

- тип волокон: Multimode (MM);
- кількість волокон: 2 (duplex);
- діаметр волокон: 50/125 мкм або 62.5/125 мкм;
- тип оболонки: PVC (Polyvinyl Chloride) або LSZH (Low Smoke Zero Halogen);

- довжина кабелю: Від 1 метра до кількох кілометрів, в залежності від потреб мережі.

Рисунок 2.11 – Кабель 50/125 Duplex Fiber Patch Cable [43]

2.4 Види архітектур мультикомп’ютерної системи

2.4.1 Кластерна архітектура

Кластерне обчислення відображає систему (рисунок 2.12), яка складається з двох або більше комп'ютерів або систем, часто відомих як вузли. Ці вузли співпрацюють для виконання програм та виконання інших завдань. Користувачі, які використовують вузли, мають сприйняття того, що до них відповідає лише одна система, створюючи ілюзію єдиного ресурсу, відомого як віртуальні машини. Цей концепт визначається як прозорість системи. Інші важливі функції, необхідні для побудови таких платформ, включають надійність, балансування навантаження та продуктивність [44-48].

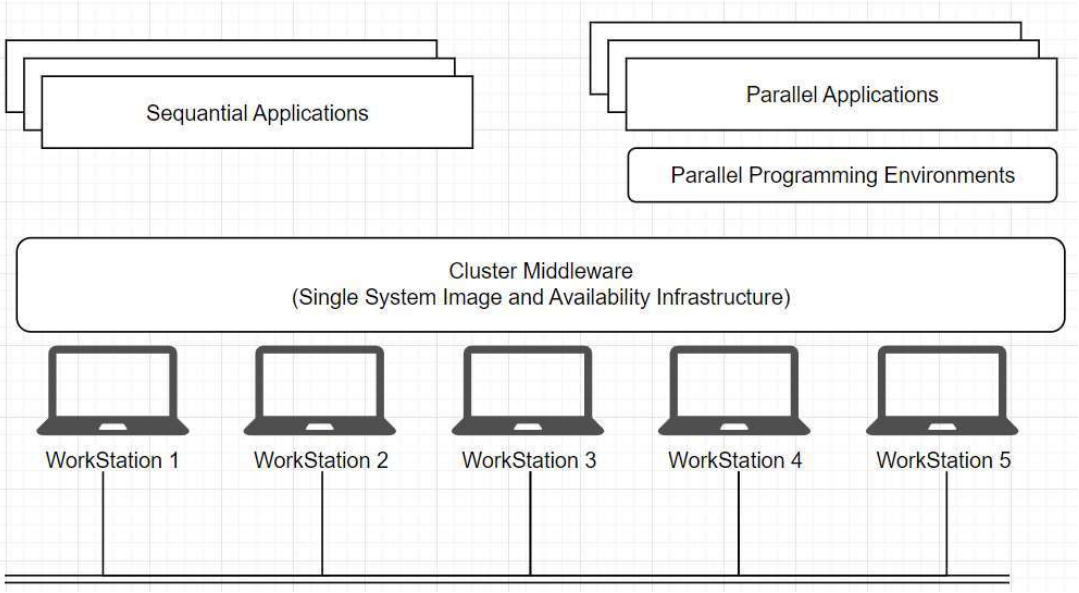


Рисунок 2.12 – Архітектура кластерного обчислення

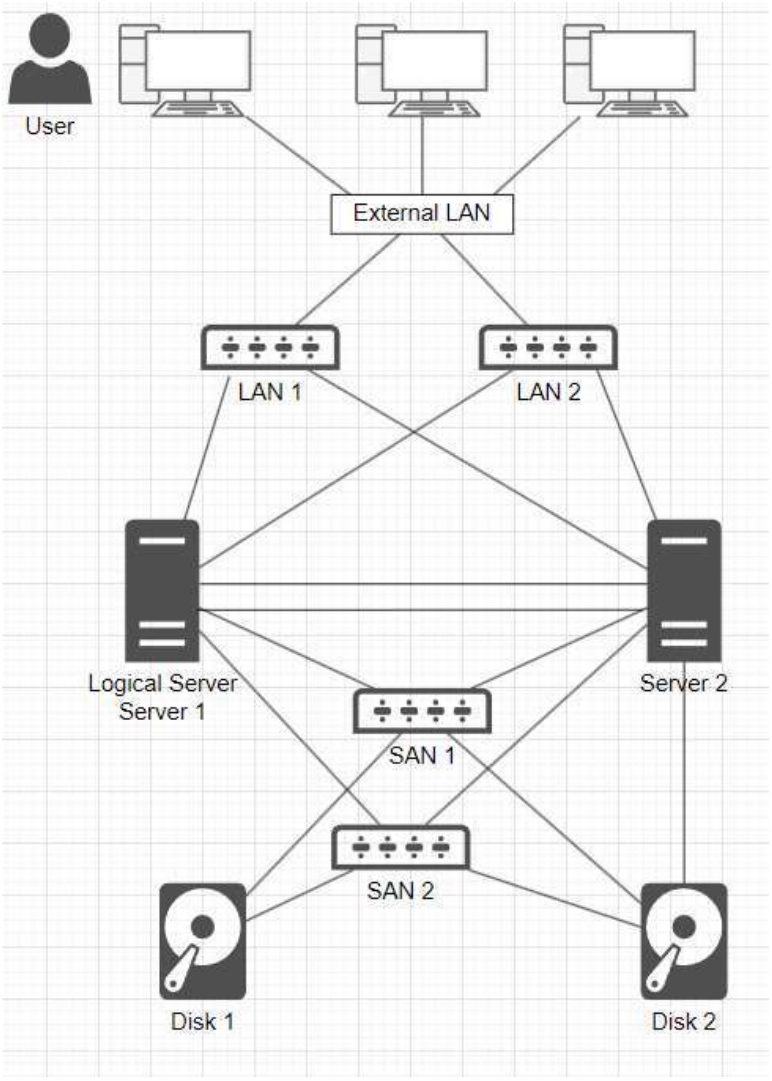


Рисунок 2.13 – Архітектура високої доступності

Висока доступність (HA) та відмовостійкі кластери (рисунок 2.13) – ці моделі кластерів забезпечують безперервну доступність сервісів та ресурсів завдяки вбудованій надлишковості системи. Основна ідея цієї форми кластеру полягає в тому, що якщо один вузол виходить з ладу, то додатки та сервіси можуть бути доступні на інших вузлах. Ці типи кластерів служать основою для критичних завдань, поштових сервісів, файлових серверів та серверів додатків.

Кластери для балансування навантаження (рисунок 2.14) – ці кластери розподіляють вхідний трафік або запити на ресурси між вузлами, які виконують однакові програми та працюють на тих самих машинах. У цій кластерній моделі всі вузли обробляють запити, і в разі виходу з ладу одного з них, запити розподіляються між усіма іншими доступними вузлами. Такі рішення часто використовуються у веб-серверних фермах.

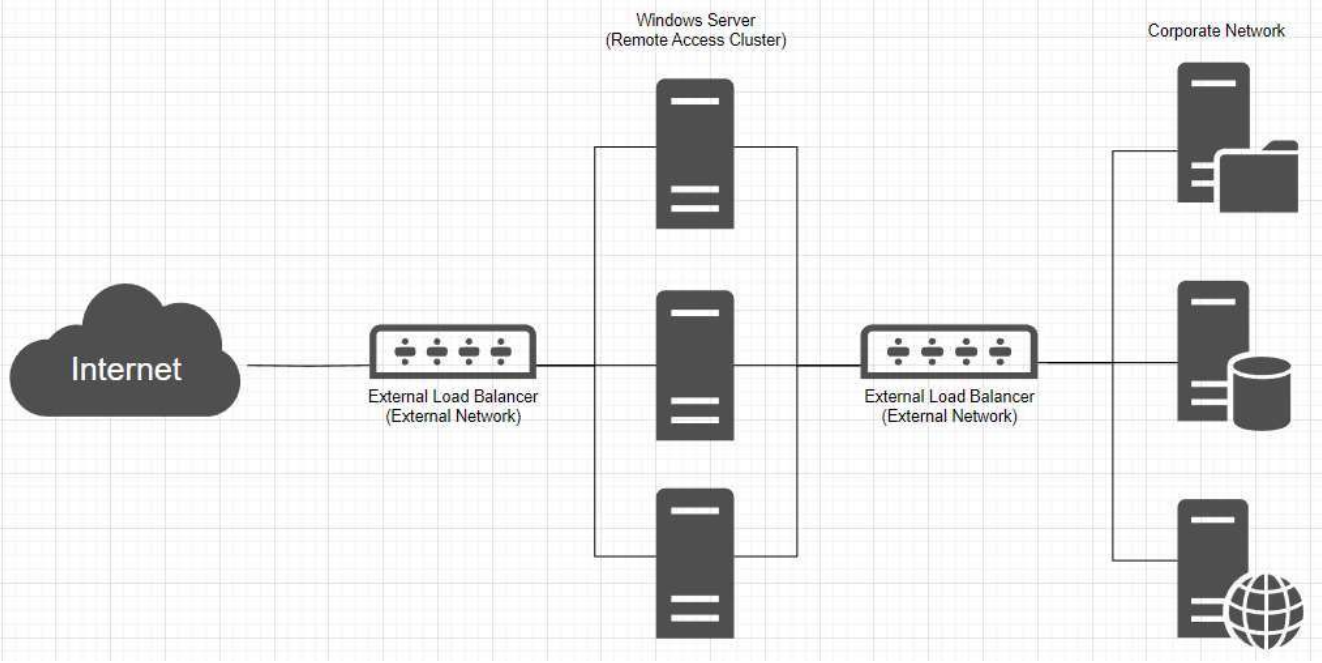


Рисунок 2.14 – Архітектура кластерного балансування

Кластери високої доступності та балансування навантаження (рисунок 2.15) – ця модель кластерів об'єднує функції обох типів кластерів, забезпечуючи підвищену доступність та масштабованість сервісів і ресурсів. Такі кластери часто використовуються для поштових, веб-, новинних і FTP серверів.

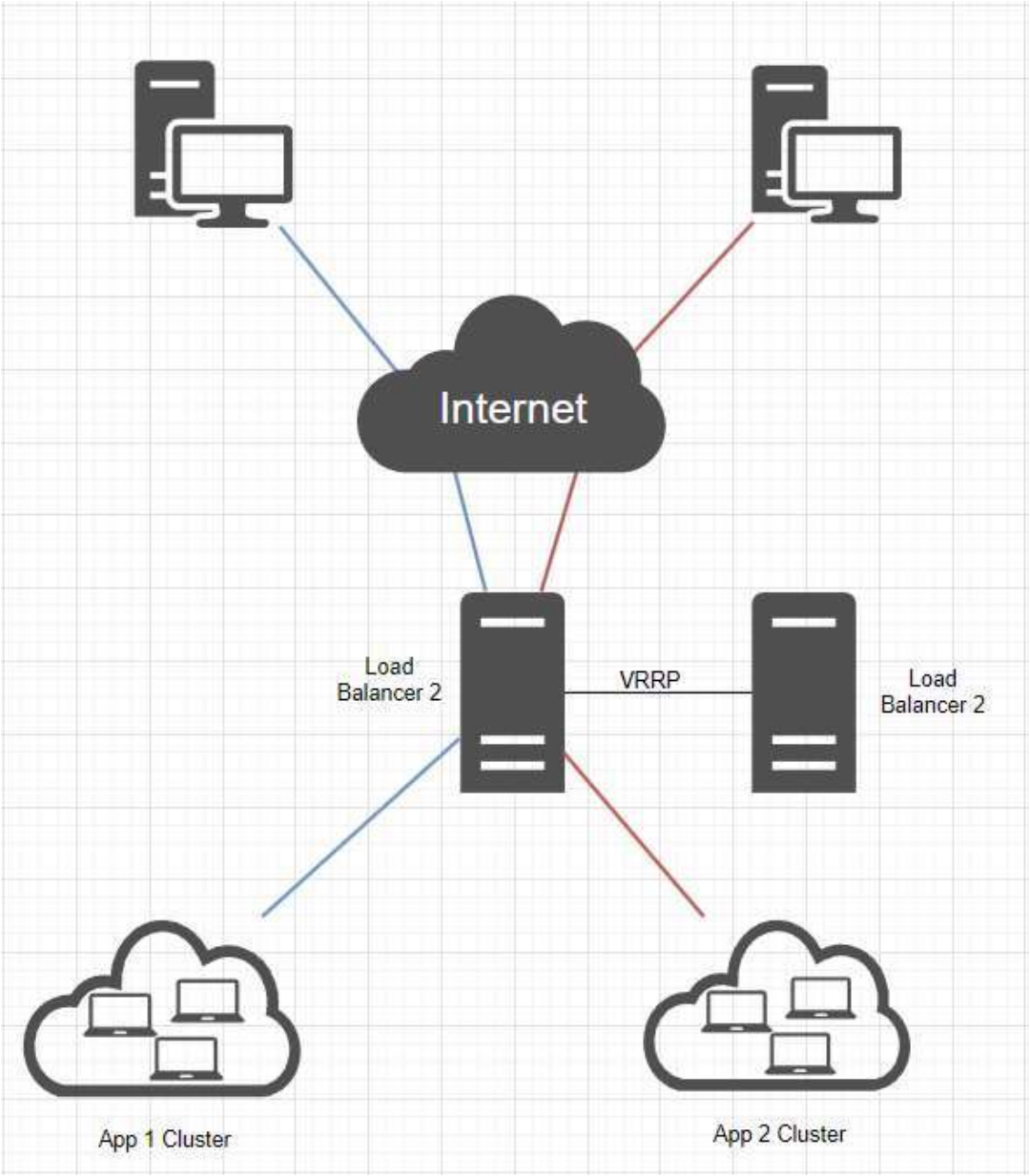


Рисунок 2.15 – Архітектура кластерів високої доступності

Кластери розподіленої та паралельної обробки (рисунок 2.16) – ця модель кластерів підвищує доступність і продуктивність для застосунків з великими обчислювальними завданнями. Велике обчислювальне завдання розділяється на менші частини і розподіляється між станціями. Такі кластери зазвичай використовуються для наукових обчислень або фінансового аналізу, що потребують високої обчислювальної потужності.

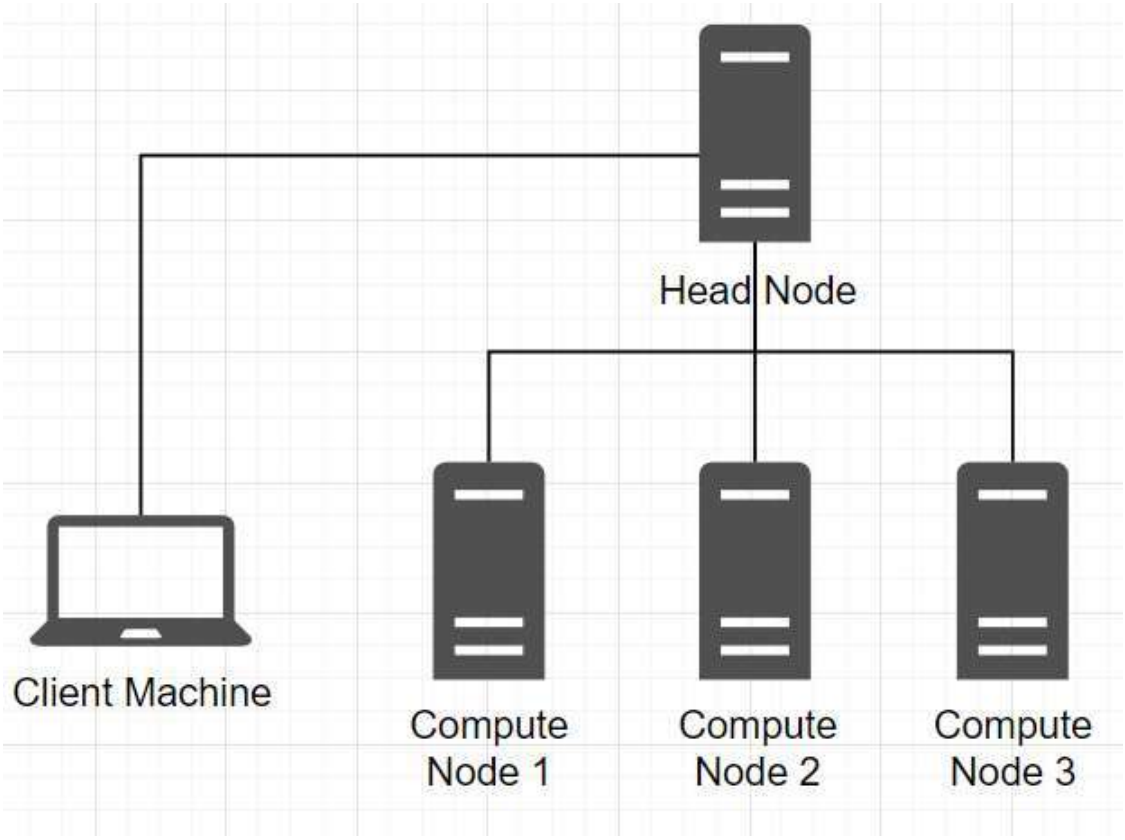


Рисунок 2.16 – Архітектура розподіленого кластеру

2.4.2 Масивно-паралельна архітектура

Масивно-паралельна архітектура (MPP) – це тип архітектури, де пам'ять фізично розділена (рисунок 2.17). У цій конфігурації система складається з окремих модулів, кожен з яких містить процесор, локальний банк оперативної пам'яті (ОЗП), два комунікаційних процесора (роутера) або мережевий адаптер, іноді - жорсткі диски або інші пристрої введення/виводу. Один роутер використовується для передачі команд, а інший – для передачі даних. З суті, такі модулі представляють собою повнофункціональні комп'ютери [49].

Процесори MPP можуть мати до 200 або більше процесорів, які працюють над додатком та, зазвичай, взаємодіють за допомогою інтерфейсу обміну повідомленнями. MPP працює, дозволяючи надсилати повідомлення між процесами шляхом використання «взаємозв'язку» для розташування шляхів даних

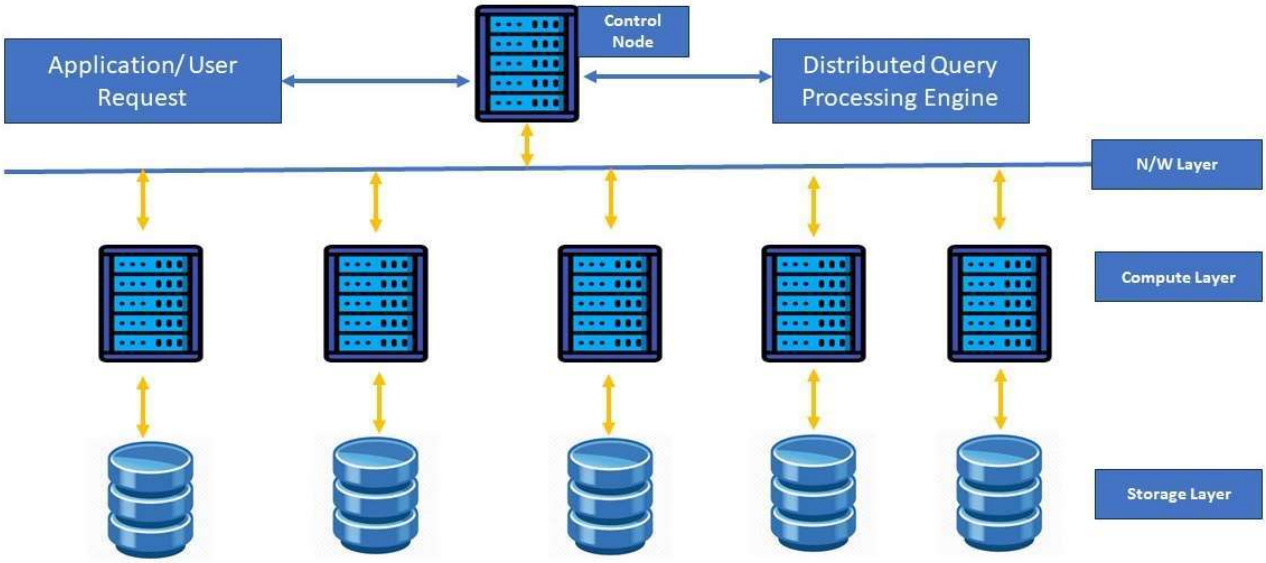


Рисунок 2.17 – Схема масивно паралельної архітектури [50]

Існує кілька типів архітектур баз даних MPP, кожна з яких має свої переваги:

- грід-обчислення – це використання кількох комп'ютерів у розподілених мережах. У цьому типі архітектури ресурси використовуються опортуністично в залежності від їх доступності. Ця архітектура зменшує витрати на серверний простір, але також обмежує пропускну здатність у часи пік або коли надходить забагато запитів;
- комп'ютерна кластеризація – це об'єднання доступної потужності у вузли, які можуть з'єднуватися один з одним для виконання кількох завдань одночасно.

Переваги архітектури MPP:

- продуктивність – швидкість обчислень зростає лінійно. Чим більше вузлів, тим швидше виконуються агрегації та обчислення для всього набору даних;
- масштабованість – ми можемо масштабувати систему необмежено. Додавши більше вузлів до нашої архітектури, ми можемо розширити нашу базу даних MPP для зберігання та обробки більших обсягів даних;
- вартість – оскільки ми додаємо більше вузлів, легше обробляти більше даних за допомогою менш вартісного обладнання;

- відсутність одного точки відмови: якщо один вузол вийде з ладу, інші вузли все ще можуть працювати та підтримувати діяльність бази даних під час його обслуговування;
- еластичність: якщо я хочу додати більше вузлів до бази даних, це легко зробити без недоступності всього кластеру.

2.4.3 Симетрична багатопроцесорна архітектура

Симетрична багатопроцесорна архітектура (SMP) – це тип архітектури, де всі процесори мають спільний доступ до фізичної пам'яті системи (рисунок 2.18).

Пам'ять служить засобом передачі повідомлень між процесорами, при цьому всі обчислювальні пристрої мають рівні права доступу до неї і однакові адреси для всіх областей пам'яті. Тому SMP архітектура отримала назву "симетрична". Ця особливість дозволяє ефективно обмінюватися даними між різними обчислювальними пристроями.

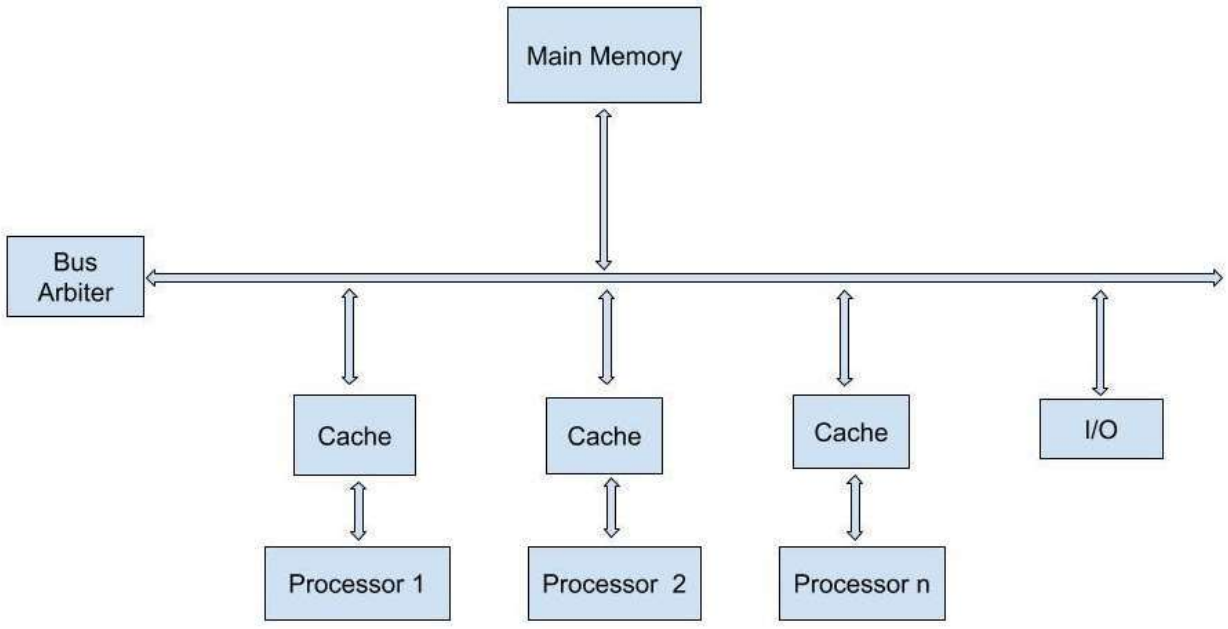


Рисунок 2.18 – Схема багатопроцесорної архітектури

Система на базі SMP будується на основі швидкісної системної шини (наприклад, SGI PowerPath, Sun Gigaplane, DEC TurboLaser), до якої підключаються функціональні блоки трьох типів: процесори (ЦП), операційна система (ОС) та підсистема вводу/виводу (I/O). Для підключення модулів I/O використовуються більш повільні шини (PCI, VME64). Найвідомішими SMP-системами є машини серій SGI Power Chalenge, DEC Alpha Server, Cray T3D. Вся система працює під управлінням однієї ОС (зазвичай UNIX-подібної, але для платформи Intel підтримується Windows NT). ОС автоматично (під час роботи) розподіляє процеси між процесорами, але іноді можлива явна прив'язка.

Переваги SMP-систем:

- простота та універсальність для програмування;
- легкість експлуатації;
- відносно невисока ціна.

З недоліків, це – система з загальною пам'яттю, побудовані на системній шині, погано масштабуються.

2.4.4 Вибір архітектури мультикомп’ютерної системи

На сьогоднішній день майже всі великі програмні системи є розподіленими. Розподіленою називається така система, в якій обробка інформації виконується не на одному комп'ютері, а розподіляється між кількома комп'ютерами. Хоча проектування розподілених систем має багато спільного з розробкою будь-якого іншого програмного забезпечення, все ж варто враховувати деякі специфічні особливості. Деякі з цих аспектів вже були розглянуті у вступі до розділу 10, де обговорювалася архітектура клієнт/сервер, але тут вони обговорюються детальніше.

З огляду на широке поширення розподілених систем у наші дні, розробники програмного забезпечення повинні бути обізнані з їх специфікою. До недавнього часу всі великі системи були переважно централізованими, запускаючись на одному головному комп'ютері (мейнфреймі) з підключеними до нього

терміналами. Термінали майже не виконували обробку інформації – всі обчислення виконувалися на головному комп'ютері. Розробникам таких систем не доводилося турбуватися про проблеми розподілених обчислень.

Сучасні програмні системи можна поділити на три великі класи:

1. Прикладні програмні системи, призначені для роботи на одному персональному комп'ютері або робочій станції. До них належать текстові процесори, електронні таблиці, графічні системи тощо.

2. Вбудовані системи, призначені для роботи на одному процесорі або на інтегрованій групі процесорів. Це системи управління побутовими пристроями, різними приладами та інше.

3. Розподілені системи, в яких програмне забезпечення виконується на слабо інтегрованій групі паралельно працюючих процесорів, з'єднаних через мережу. До них належать системи банкоматів банків, видавничі системи, системи колективного користування програмним забезпеченням тощо.

Завданням розробки розподілених систем є створення програмного або апаратного забезпечення з необхідними характеристиками розподіленої системи. Для цього важливо знати переваги та недоліки різних архітектур розподілених систем. Тут виділяють два основних типи архітектури розподілених систем:

1. Архітектура клієнт/сервер. У цій моделі систему можна уявити як набір сервісів, що надаються серверами клієнтам. В таких системах сервери та клієнти суттєво відрізняються один від одного.

2. Архітектура розподілених об'єктів. У цьому випадку немає відмінностей між серверами і клієнтами, і систему можна уявити як набір взаємодіючих об'єктів, чиє місцеположення не має значення. Відмінності між постачальниками сервісів і їх користувачами відсутні.

У розподіленій системі різні компоненти можуть бути реалізовані на різних мовах програмування та виконуватися на різних типах процесорів. Моделі даних, представлення інформації та протоколи взаємодії в такій системі можуть бути неоднорідними. Тому для розподілених систем необхідне програмне забезпечення,

яке могло б керувати цими різнорідними компонентами і забезпечувати їхню взаємодію та обмін даними. Проміжне програмне забезпечення належить саме до такого типу ПО, оскільки воно функціонує як посередник між різними частинами розподіленої системи.

2.5 Висновки

У цьому розділі оцінено та розглянуто різні типи топологій в мультикомп'ютерних системах. В ході аналізу серед трьох типів топологій, відносно їх використання, переваг та недоліків для проектування обрано саме топологію “кільце”. Через те, що топологія “кільце” є найбільш вразливою до пошкоджень кабелю, серед трьох розглянутих топологій, вирішено передбачити прокладання двох паралельних ліній зв'язку, одна з яких є резервною, але у поточні архітектурі вона буде забезпечувати передачу даних у двох напрямках.

Таким чином, утворюється нова топологія “кільце” двонаправленого зв’язку. Мета такого підходу - подвоїти швидкість передачі даних у мережі. При пошкодженні одного з кабелів мережа може переключитися на інший кабель, хоча максимальна швидкість передачі буде меншою. Топологія “кільце” двонаправленого зв’язку має свої переваги та недоліки.

Для побудови мультикомп'ютерної системи для обраної топології розглянуто та обрано необхідне апаратне обладнання. що включає сервер, клієнтські комп'ютери, комутатори, маршрутизатори та кабелі. Для побудови ефективної мережі обрано сервер Dell PowerEdge T340, комп'ютери HP ProDesk 600 G6 Microtower для клієнтів системи, комутатори Cisco Catalyst 2960 Series та маршрутизатори Cisco ISR 4000 Series. Для передачі та з'єднання компонентів системи зручним та оптимальним способом обрано волоконно-оптичні кабелі.

Також було розглянуто різні архітектури мультикомп’ютерних систем, які можуть бути застосовані у залежності від потреб проекту та специфікацій вимог. Серед архітектур виділено кластерну архітектуру, масивно-паралельну архітектуру (MPP) та симетричну багатопроцесорну архітектуру (SMP). Кластерні системи

надають можливість забезпечення високої доступності та масштабованості, що особливо важливо для систем з високими вимогами до надійності та продуктивності. Масивно-паралельні системи забезпечують високу продуктивність за рахунок паралелізму обробки даних і є оптимальними для завдань, які потребують інтенсивних обчислень. Симетричні багатопроцесорні системи пропонують легкість управління та налаштування, а також гнучкість у розширенні ресурсів.

3 ПРОЕКТУВАННЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ

3.1 Алгоритм об’єданння складових мультикомп’ютерної системи

Розглянемо послідовність для підключення пристроїв для топології «кільце» мультиком’ютерної системи із двнонаправленим зв’язком. Даний алгоритм виглядатиме так:

Підготовка обладнання:

- підключіть мережеві адаптери до кожного комп’ютера;
- використовуйте кабелі для створення кільцевої або подвійної кільцевої топології між комп’ютерами.

Підключення комутаторів:

- для подвійного кільця підключіть перший і другий комутатор до кожного з комп’ютерів за допомогою двох кабелів;

Підключення маршрутизаторів (якщо потрібно):

- можуть бути підключені до будь-якого з комп’ютерів у кільці або комутатора для доступу до інших мереж або Інтернету.

Налаштування програмного забезпечення:

- налаштуйте IP-адреси для мережевих адаптерів та маршрутизаторів;
- встановіть необхідне програмне забезпечення для управління мережевими пристроями.

Перевірка з'єднання:

- використовуйте інструменти для мережевої діагностики, такі як пакети Ping, для перевірки з'єднань між комп’ютерами;
- кожна з цих топологій забезпечує свої переваги, такі як надійність та можливість масштабування.

Після завершення під’єднання всіх компонентів схема виглядатиме як на рисунку 3.1.

Основною перевагою двонаправленого кільцевого зв'язку є підвищена надійність системи, оскільки вона може продовжувати функціонувати навіть при

виході з ладу одного з комп'ютерів. Дані в такій системі можуть обходити пошкоджену ділянку у двох напрямках, значно мінімізуючи вплив на роботу системи. Двонаправлений зв'язок також сприяє низькій затримці передачі даних, оскільки дані можуть передаватися найкоротшим шляхом до вузла призначення, забезпечуючи швидкість та ефективність.

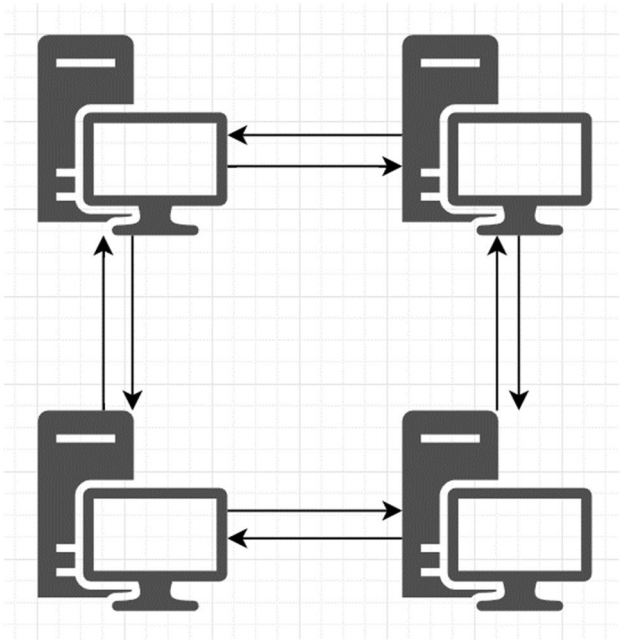


Рисунок 3.1 – Схема з’єднання компонентів для клієнтів

Легкість розширення – це ще одна ключова характеристика двонаправленої кільцевої топології. Додавання нового комп'ютера в систему здійснюється шляхом його інтеграції між існуючими вузлами без перебудови всієї мережі, що дозволяє легко масштабувати систему відповідно до зростаючих потреб.

Проте двонаправлений зв'язок також має потенційні недоліки. Хоча ризик зупинки всієї мережі через одноточковий збій знижується, необхідність у резервних механізмах та складності у діагностиці помилок залишається. Виявлення та усунення помилок вимагає більш складного моніторингу, оскільки потрібно враховувати можливість передачі даних у двох напрямках.

Використання проміжного програмного забезпечення для управління зв'язком між вузлами важливе і в двонаправленій топології. Таке програмне забезпечення допомагає абстрагуватися від технічних деталей мережевої взаємодії,

пропонуючи розробникам більш простий і гнучкий інтерфейс для керування мережею. Воно може включати функції автоматичного відновлення з'єднань, управління потоками даних і балансування навантаження. Підтримка стандартних мережевих протоколів, таких як TCP/IP, є ключовою для забезпечення сумісності і інтероперабельності з іншими системами і додатками.

3.2 Вибір засобів розробки проміжного ПЗ

C++ і мультикомп'ютерні системи мають тісний зв'язок завдяки численним перевагам, які C++ надає для розробки таких систем [51, 52].

C++ є кросплатформною мовою програмування, що дозволяє створювати додатки для різних операційних систем, таких як Windows, Linux, macOS та інші. Це забезпечує гнучкість і зручність розробки мультикомп'ютерних систем, які можуть працювати в різних середовищах. C++ дозволяє розробляти високопродуктивний код, що є критичним для мультикомп'ютерних систем, де швидкість обробки даних і ефективне використання ресурсів мають вирішальне значення.

Однією з головних переваг C++ є детальний контроль над управлінням пам'яттю. Це дозволяє уникати проблем з витоками пам'яті і забезпечувати стабільну роботу системи навіть при великих навантаженнях. Розробники можуть ефективно використовувати динамічний розподіл пам'яті для обробки великих обсягів даних і забезпечення швидкого доступу до них. Надає розробникам детальний контроль над управлінням пам'яттю, що дозволяє оптимізувати використання ресурсів і забезпечувати стабільність роботи системи навіть при високих навантаженнях [53-56].

C++ має потужні засоби для роботи з багатопоточністю, що дозволяє ефективно використовувати можливості сучасних багатоядерних процесорів для забезпечення одночасної обробки численних запитів.

Існує багато бібліотек і фреймворків для C++, які спрощують розробку мережевих додатків та обміну даними між комп'ютерами. Наприклад, Boost.Asio,

gRPC і ZeroMQ забезпечують зручні інструменти для реалізації двонаправленого зв'язку [57, 58].

C++ є кросплатформною мовою програмування, що дозволяє розробляти додатки для різних операційних систем, таких як Windows, Linux та macOS, з мінімальними змінами в коді.

Завдяки великій і активній спільноті розробників, доступна багата документація, численні приклади та підтримка, що спрощує розробку та усунення проблем.

Для C++ програм існують два основні методи зберігання інформації в оперативній пам'яті комп'ютера. Перший метод полягає у використанні змінних. Пам'ять, виділена для змінних, закріплюється за ними під час компіляції та не може бути змінена під час виконання програми. Другий метод використовує систему динамічного розподілу пам'яті в C++. У цьому випадку пам'ять для даних виділяється за потреби з розділу вільної пам'яті, що розташований між вашою програмою (та її постійною областю зберігання) і стеком. Динамічне виділення пам'яті означає отримання програмою пам'яті під час її виконання. Іншими словами, завдяки цій системі програма може створювати змінні під час виконання, залежно від потреби. Така система особливо корисна для структур даних, як-от зв'язні списки та двійкові дерева, які змінюють свій розмір під час використання.

Динамічне виділення пам'яті є важливою складовою майже всіх реальних програм. Щоб задовольнити запит на динамічне виділення пам'яті, використовується так звана "купа". Однак, у деяких надзвичайних ситуаціях вільна пам'ять у купі може вичерпатися. Отже, хоча динамічний розподіл пам'яті забезпечує більшу гнучкість порівняно з фіксованим розподілом, він також має свої межі використання.

Сокети відіграють важливу роль у програмуванні, оскільки вони забезпечують можливість взаємодії між пристроями в мережі. Це особливо важливо для багатьох програм, таких як чати, відеоконференції та додатки для передачі файлів [59].

Крім того, сокети дозволяють створювати клієнт-серверні програми. У таких програмах сервер надає певні послуги, а клієнт їх використовує. Наприклад, веб-сервер надає веб-сторінки, а веб-браузер виступає як клієнт, що запитує ці сторінки у сервера.

Спочатку необхідно створити сокет. Сокет є кінцевою точкою двонаправленого з'єднання, тобто об'єктом, який відповідає за управління зв'язком між двома пристроями. Сокет можна створити за допомогою функції socket().

Після створення сокета необхідно прив'язати до нього адресу. Адреса, що складається з пари IP/порт, унікально ідентифікує сокет у мережі. Для прив'язки адреси до сокета використовується функція bind().

Після створення сокета та призначення адреси необхідно встановити з'єднання з іншим пристроєм. Процес встановлення з'єднання залежить від використовуваного мережевого протоколу, але загалом включає наступні кроки:

- клієнтський сокет створює сокет і прив'язує його до IP-адреси/порту;
- клієнтський сокет надсилає запит на підключення до серверного сокета;
- серверний сокет приймає запит на підключення та створює новий сокет для зв'язку з клієнтом;
- клієнтський і серверний сокети обмінюються інформацією для встановлення параметрів підключення;
- з'єднання встановлено, і можна почати обмін даними.

Після встановлення з'єднання дані можна передавати і отримувати між пристроями через сокет. Для відправлення даних використовується функція send(), а для отримання даних – функція recv().

3.3 Архітектура розподілених об'єктів типу клієнт – сервер

В архітектурі клієнт-сервер програмне застосування розглядається як сукупність сервісів, що надаються серверами, і множину клієнтів, які користуються цими сервісами (рисунок 3.2). Клієнти повинні знати про наявні сервери, хоча можуть бути не в курсі існування інших клієнтів.

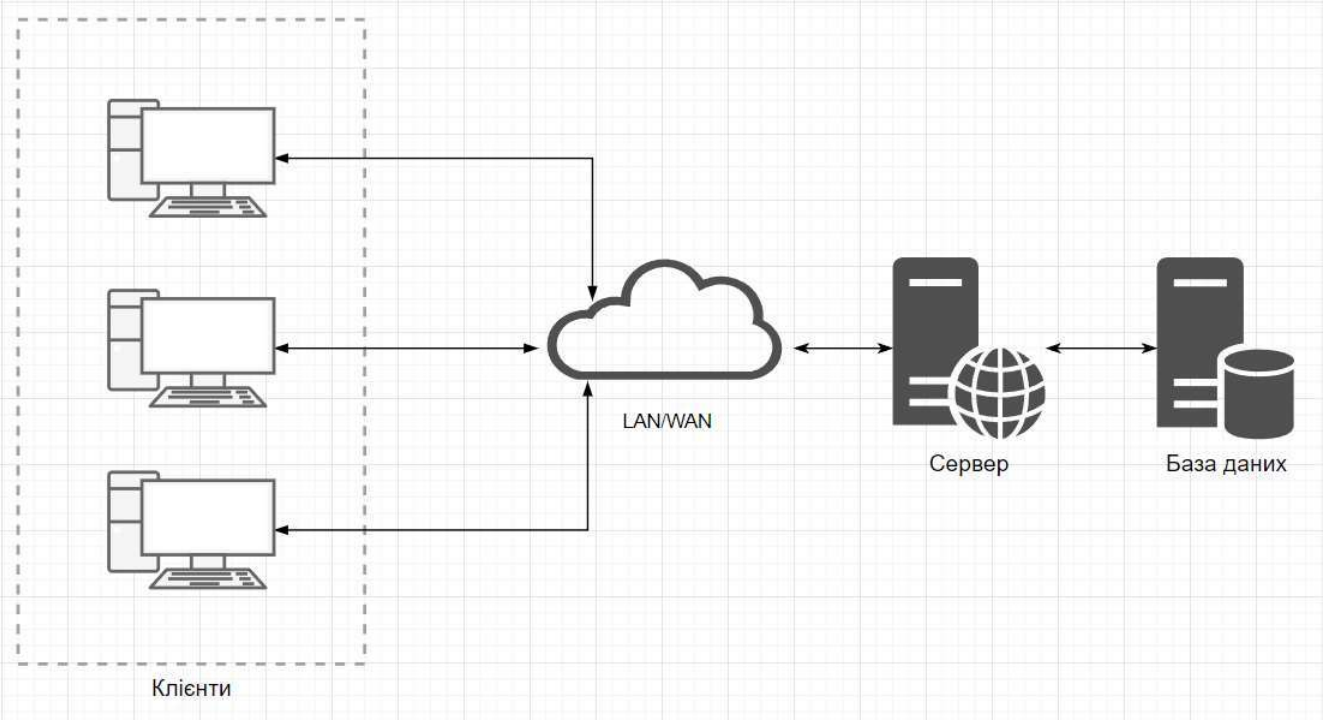


Рисунок 3.2 – Архітектура клієнт-сервер

Модель архітектури клієнт-сервер є моделлю розподіленої системи, яка демонструє розподіл даних і процесів між кількома процесорами. Вона складається з трьох основних компонентів:

1) набір автономних серверів, що надають сервіси іншим підсистемам. Наприклад, сервер друку, що забезпечує послуги друку, файлові сервери, які надають послуги з управління файлами, і сервер-компілятор, що пропонує послуги компіляції вихідного коду програм;

2) набір клієнтів, які використовують сервіси, що надаються серверами. У контексті системи клієнти є звичайними підсистемами, і кілька екземплярів клієнтської програми можуть виконуватися паралельно;

3) мережа, через яку клієнти отримують доступ до сервісів. Хоча клієнти і сервери можуть запускатися на одній машині, на практиці модель клієнт/сервер зазвичай не застосовується в такій ситуації.

Клієнти повинні знати імена доступних серверів і сервіси, які ті надають. Сервери, натомість, не зобов'язані знати ні імена клієнтів, ні їх кількість. Доступ до

сервісів, які надаються серверами, здійснюється за допомогою віддаленого виклику процедур.

Підхід клієнт/сервер можна застосовувати для реалізації систем, заснованих на репозиторії, який підтримується як сервер системи. Підсистеми, що мають доступ до репозиторію, виступають клієнтами, хоча кожна підсистема зазвичай управляє власними даними. Під час роботи сервери і клієнти обмінюються даними, проте при передачі великих об'ємів даних можуть виникати проблеми з пропускною спроможністю мережі. Однак з розвитком все швидших мереж ця проблема стає менш актуальною.

Найбільша перевага моделі клієнт/сервер полягає в її розподіленій архітектурі. Ця модель ефективно працює в мережевих системах з багатьма розподіленими процесорами. Додавання нового сервера в систему або його інтеграція з іншими частинами системи є простим, і оновлення серверів не впливає на інші частини системи.

3.4 Архітектура програмного забезпечення

Клієнт-серверна архітектура з топологією двонаправленого зв'язку являє собою сучасний підхід до розробки програмного забезпечення, який забезпечує інтерактивну і гнучку взаємодію між клієнтом і сервером. У цій архітектурі обидві сторони можуть ініціювати передачу даних, що дозволяє досягти високої швидкості реакції системи та підвищення її ефективності[44].

Клієнт-серверна архітектура з двонаправленим зв'язком складається з двох основних компонентів: клієнта і сервера (рисунок 3.3). Клієнт виконує роль ініціатора запитів і отримувача відповідей, тоді як сервер обробляє ці запити та надсилає відповіді. Основна відмінність двонаправленої архітектури полягає в тому, що сервер також може ініціювати зв'язок з клієнтом, що забезпечує більш динамічну взаємодію.

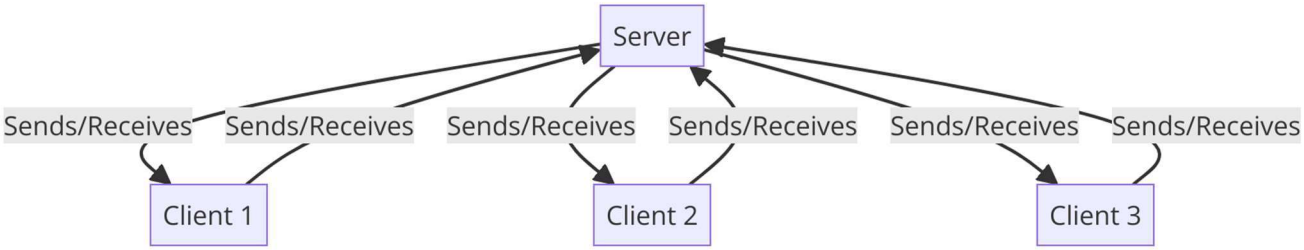


Рисунок 3.3 – Діаграма під'єднання пристроїв мережі

Одним із найбільш поширених протоколів для реалізації двонаправленого зв'язку є WebSocket. Цей протокол забезпечує постійне з'єднання між клієнтом і сервером через один TCP-порт. Це дозволяє уникнути повторної аутентифікації та встановлення нових з'єднань, що значно зменшує затримки та підвищує продуктивність системи.

Іншим протоколом, що може використовуватися для двонаправленого зв'язку, є HTTP/2. Він підтримує функцію серверних подій (Server-Sent Events, SSE), яка дозволяє серверу надсилати оновлення клієнту без необхідності отримувати запит. Це робить HTTP/2 підходящим для реалізації реальних час оновлень, хоча і з дещо меншими можливостями порівняно з WebSocket.

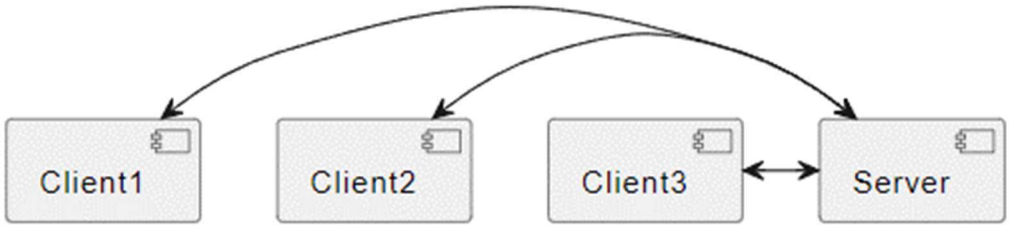


Рисунок 3.4 – Схема підключення пристроїв до серверу

Двонаправлений зв'язок реалізується за допомогою протоколів, які підтримують постійне з'єднання, таких як WebSocket. Цей протокол дозволяє встановити тривале TCP-з'єднання між клієнтом і сервером, через яке обидві сторони можуть надсилати дані в будь-який час. Такий підхід дозволяє уникнути затримок, пов'язаних із встановленням нових з'єднань, і значно підвищує ефективність обміну даними.

Серед основних переваг клієнт-серверної архітектури з двонаправленим зв'язком є:

- миттєва взаємодія між компонентами системи – це особливо важливо для додатків, де потрібна оперативна обробка даних, таких як чат-додатки або платформи для моніторингу в реальному часі;
- ефективність – постійне з'єднання зменшує затримки, пов'язані зі встановлення нового з'єднання;
- гнучкість – підтримка різних типів додатків та пристроїв, від чату на пристрої сматфону до пристроїв безпеки та датчиків.

Таким чином, наприклад, у чат-додатках клієнт надсилає повідомлення до сервера, який обробляє його і миттєво передає іншим клієнтам, забезпечуючи негайне сповіщення про нові повідомлення.

Крім того, така архітектура дозволяє підвищити гнучкість системи. Оскільки сервер може ініціювати передачу даних, з'являється можливість реалізовувати функції, які раніше були недоступні в традиційних архітектурах запит-відповідь. Наприклад, біржові платформи можуть оперативно оновлювати дані про курси валют, що дозволяє користувачам миттєво реагувати на зміни ринку [61, 62].

Проте, клієнт-серверна архітектура з двонаправленим зв'язком має і свої недоліки. Вона вимагає більше ресурсів для підтримки постійних з'єднань, що може створювати додаткове навантаження на сервер. Крім того, реалізація та підтримка такої архітектури є складнішою порівняно з традиційною архітектурою, що потребує більшої кваліфікації розробників та ретельного планування.

Клієнт-серверна архітектура з двонаправленим зв'язком є ефективним і гнучким підходом до розробки програмного забезпечення, що дозволяє забезпечити високий рівень інтерактивності та оперативності взаємодії між клієнтом і сервером. Використання сучасних протоколів, таких як WebSocket, забезпечує високу швидкість обміну даними та мінімальні затримки, що робить цю архітектуру оптимальним вибором для багатьох сучасних додатків.

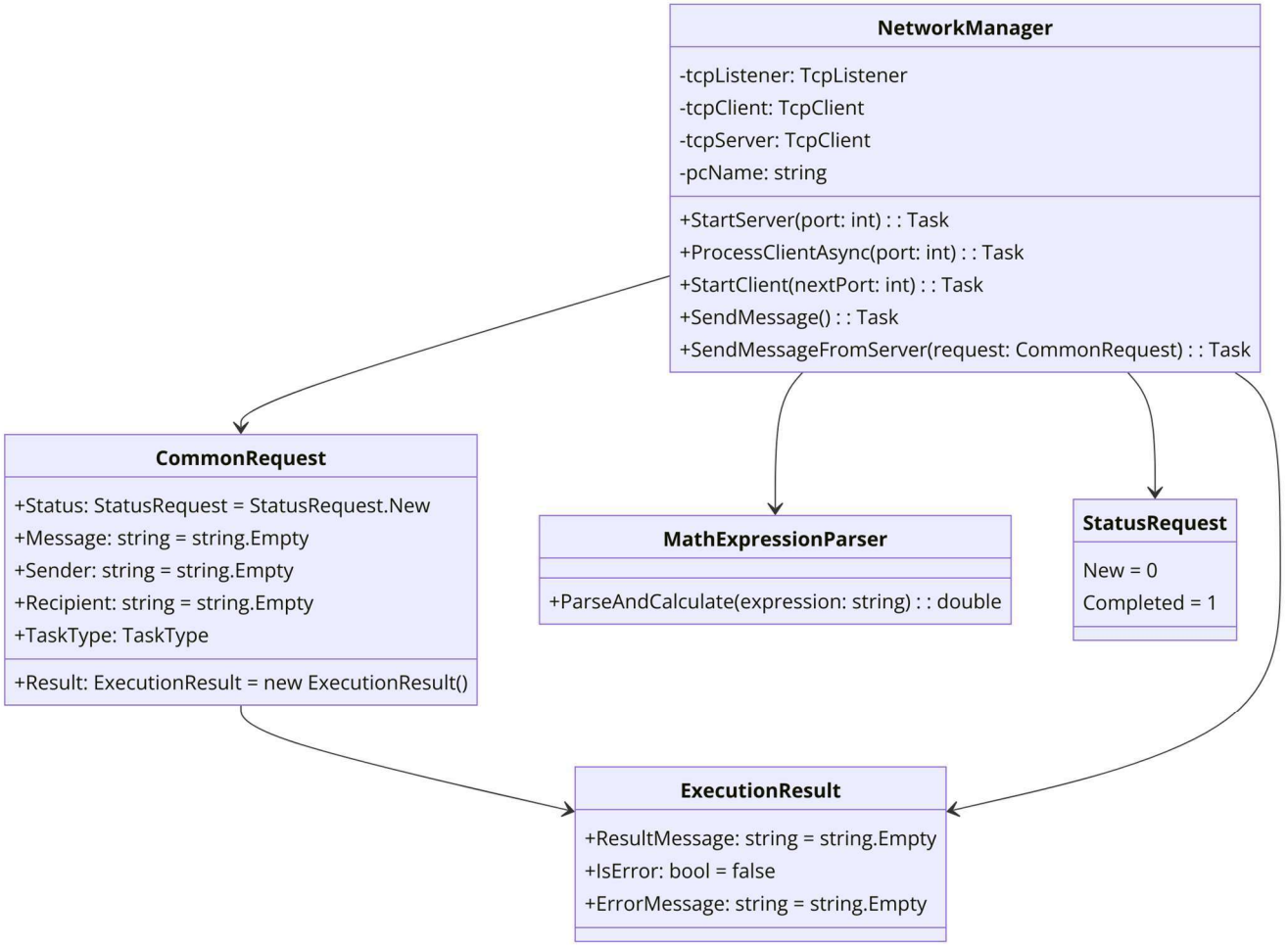


Рисунок 3.4 – Діаграма класів

У представленій діаграмі на рисунку 3.4 відображені чотири класи, кожен з яких виконує важливу роль у структурі мережевої системи.

- клас NetworkManager призначений для управління мережевими з'єднаннями, містить методи для ініціації сервера та клієнта, асинхронної обробки клієнтських запитів, відправлення повідомлень та керування повідомленнями, що надходять від сервера;
- клас CommonRequest використовується як модель для повідомлень, які циркулюють у мережі, зберігає дані про статус запиту, текст повідомлення, інформацію про відправника та отримувача, тип завдання та результати виконання;
- клас MathExpressionParser було розроблено для аналізу та виконання математичних виразів, цей клас приймає на вхід рядок з виразом і повертає числовий результат його обчислення;

- клас ExecutionResult відповідає за збереження інформації про результати виконання запитів. Він містить дані про успіх чи невдачу виконання, текст результату та текст помилки;
- клас StatusRequest як перерахування дозволяє відслідковувати стан запиту, чи він новий, чи вже завершений, що забезпечує належне управління потоками даних у системі. Загалом, ці класи спільно створюють комплексну систему для ефективної обробки даних і взаємодії в мережі.

Загальний алгоритм виконання коду.

Запуск – клієнти запускають програму, вказують її номер відносно якого налаштовується порт для прослуховування та отримання повідомлень. Тобто якщо пристроїв чотири, то вони матимуть ідентифікатор (ID) на проміжку 1-4 та відповідний порт на проміжку 12301-12304.

Після реєстрації пристрою в мережі, програма запускає два процеси:

- прослуховування – виконується в функції receiveMessages, яка перевіряє наявність нових повідомлень у мережі та обробляє їх необхідним чином;
- взаємодії – запитує у користувача номер пристрою, якому потрібно передати повідомлення, та вміст повідомлення.

Якщо повідомлення було передано до пристрою, виконується перевірка номеру пристрою отримувача в повідомленні. Якщо повідомлення призначене для іншого пристрою, то пристрій що його тримав передає його далі по мережі. Така дія, передачі повідомлення, буде виконувати поки повідомлення не досягне отримувача. Коли повідомлення отримано пристроєм, для його було надіслано, пристрій починає його обробку.

Для взаємодії використовується окремий процес. Після введення номеру, відбувається перевірка чи номер правильний та відповідає діапазону 1-4. Після правильного вказання номеру користувач вказує який вміст повідомлення потрібно передати.

На основі описаного алгоритму вище, було розроблено консольний застосунок, який потрібно запускати на пристроях та сервері. Сервером буде

виступити пристрій під номером 1, що має власний інтерфейс для керування. Пристрої будуть під номерами 2-4.

Запустимо програму ring_network на сервері та очікуємо появи наступного повідомлення що пристрій готовий та запитання для користувача на введення номеру пристрою для передачі (рисунок 3.5).

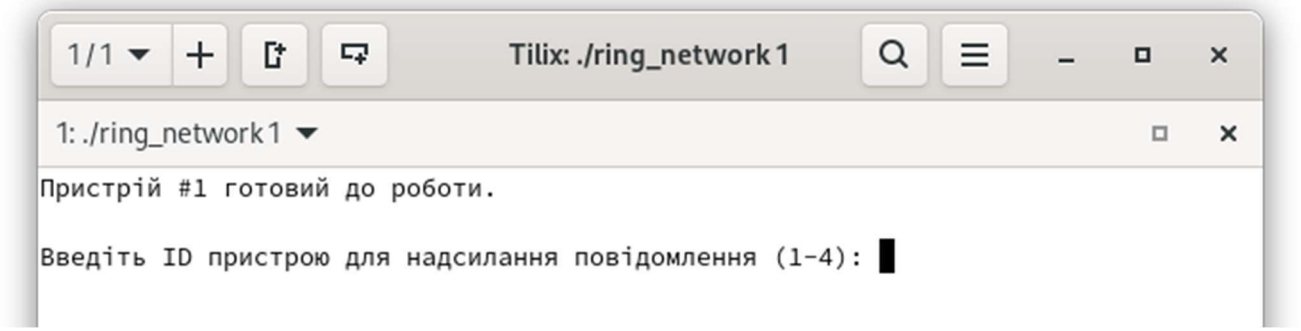


Рисунок 3.5 – Термінал на сервері із програмою

Після запуску програми на сервері встановлено сокет з'єднання в мережі для прослуховування та запущено процес для взаємодії із користувачем. Далі запустимо програму на пристрої під номером 2 (рисунок 3.6) і також очікуємо на завершення процесу налаштування мережі.

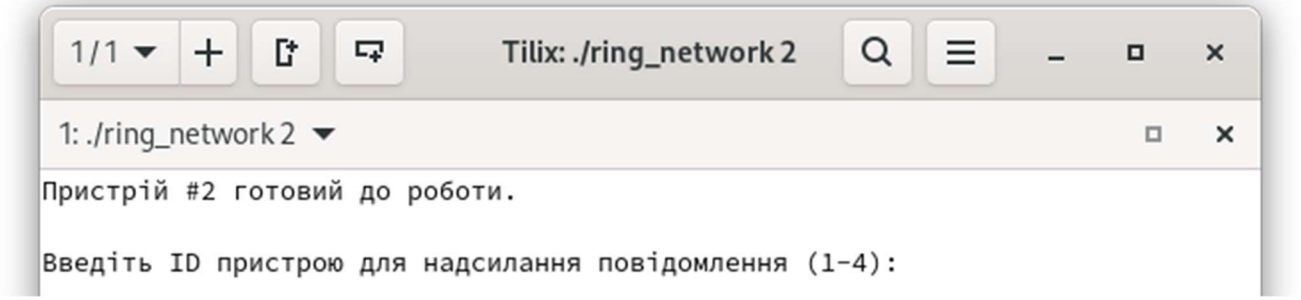


Рисунок 3.6 – Термінал на пристрої №2

Після встановлення спробуємо перевірити з'єднання із пристроєм надіславши повідомлення з серверу на пристрій за схемою на рисунку 3.7.

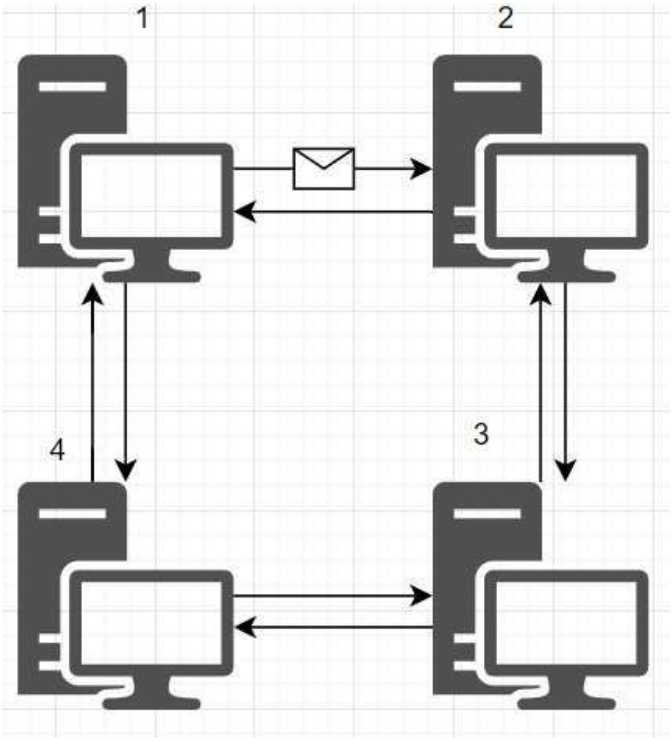


Рисунок 3.7 – Схема передачі повідомлення від серверу до пристрою №2

На рисунках 3.8а та б зображено передачу повідомлення сервером та отримання повідомлення клієнтом. Для зручності та інформативності ця інформація виводиться користувачеві.

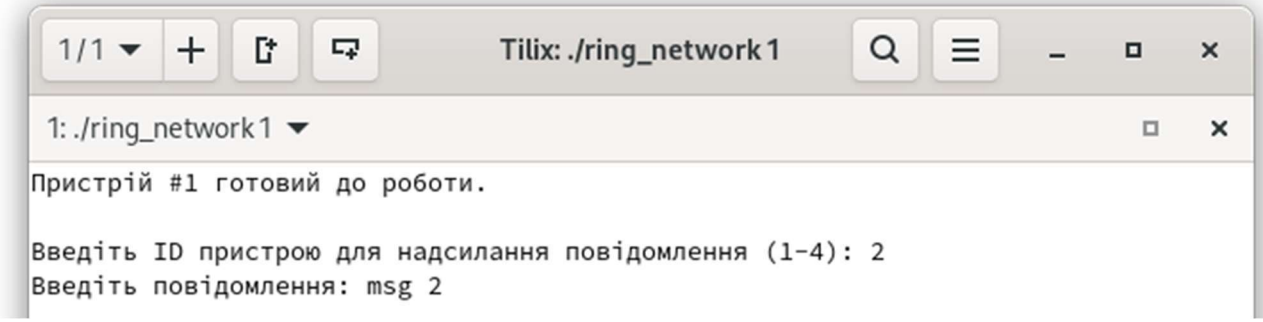


Рисунок 3.8а – Термінал серверу після надсилання повідомлення

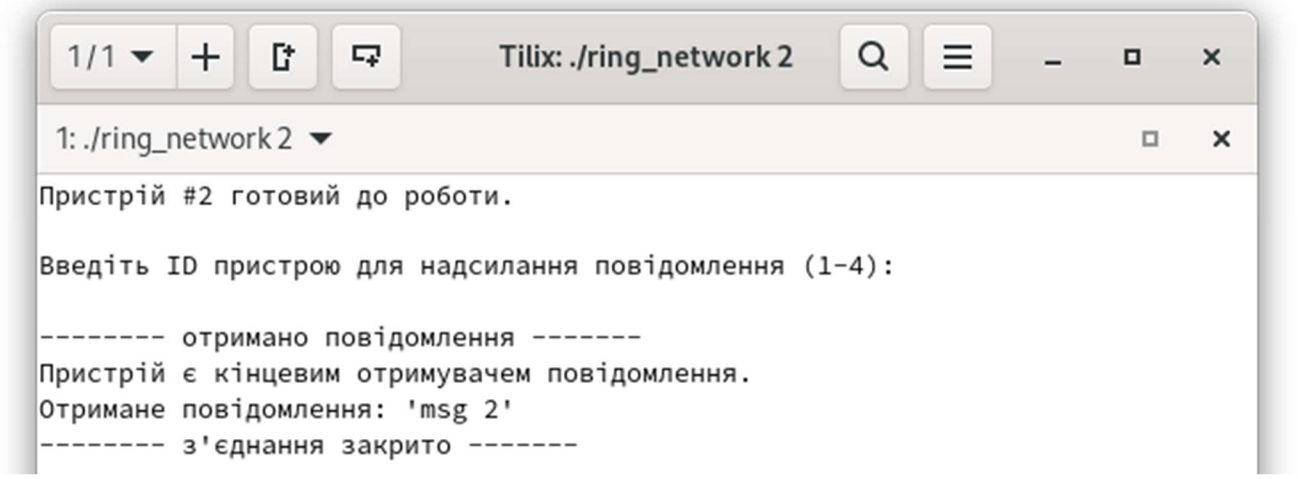


Рисунок 3.8б – Термінал пристрою №2 після отримання повідомлення

Після перевірки між двома компонентами підлюкчемо до мережі кільце ще два пристрої під номерами 3 та 4. Для цього також запустимо в них термінали та програму (рисунок 3.9).

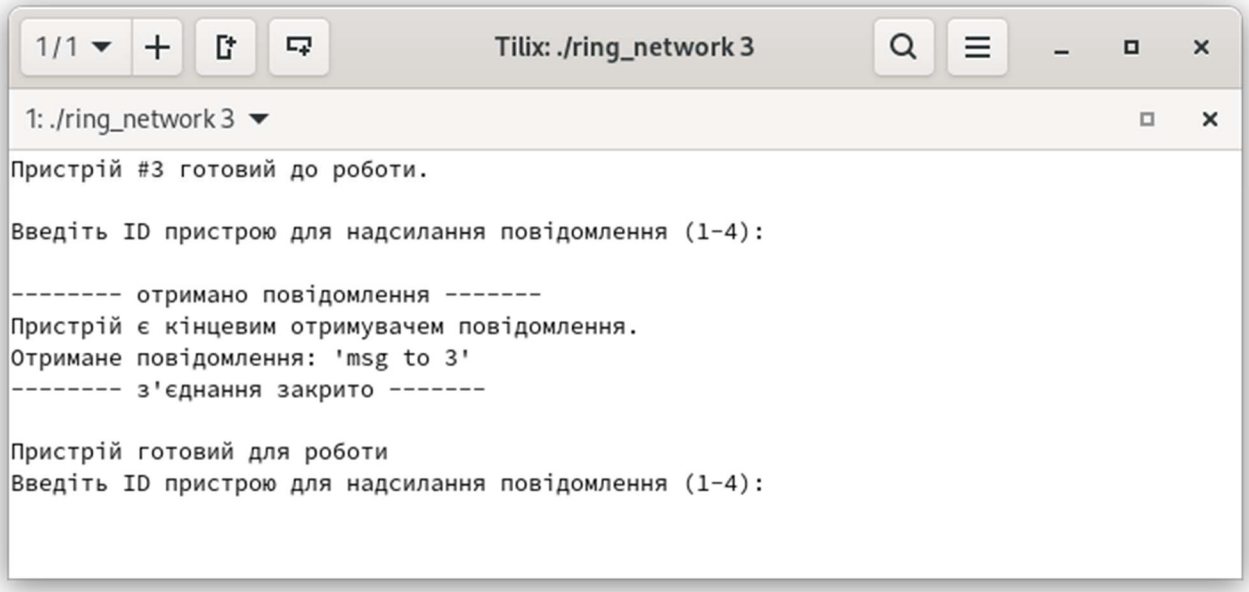


Рисунок 3.9 – Термінал на пристрої №3 після запуску

Тепер спробуємо передати повідомлення за схемою на рисунку 3.10. Надіславши від серверу №1 повідомлення до пристрою №3, повідомлення має пройти через пристрій №2.

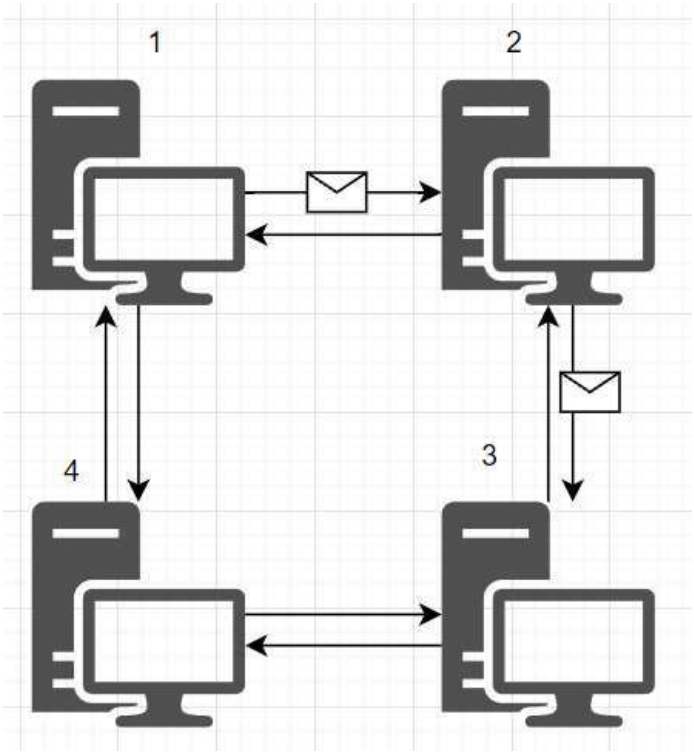


Рисунок 3.10 – Схема передачі повідомлення від серверу до пристрою №3

Для передачі на сервері введемо номер пристрою 3 та вкажемо вміст повідомлення як на рисунку 3.11.

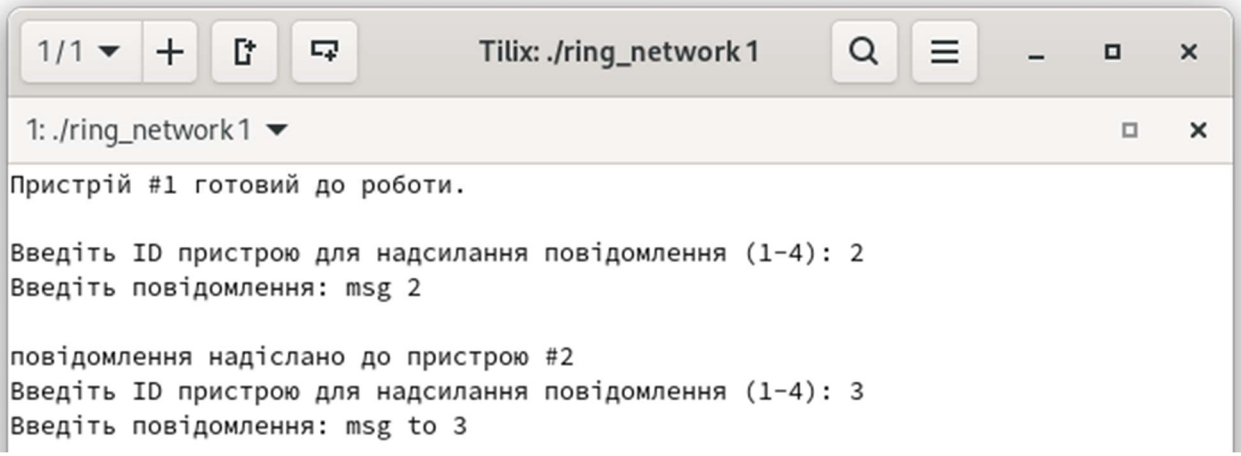


Рисунок 3.11 – Введення даних на сервері

Після надсилання бачимо на рисунку 3.12 інформацію про те, що пристрій №2 отримав повідомлення та передав його далі по мережі, оскільки кінцевий пристрій у повідомлені вказаний інший.

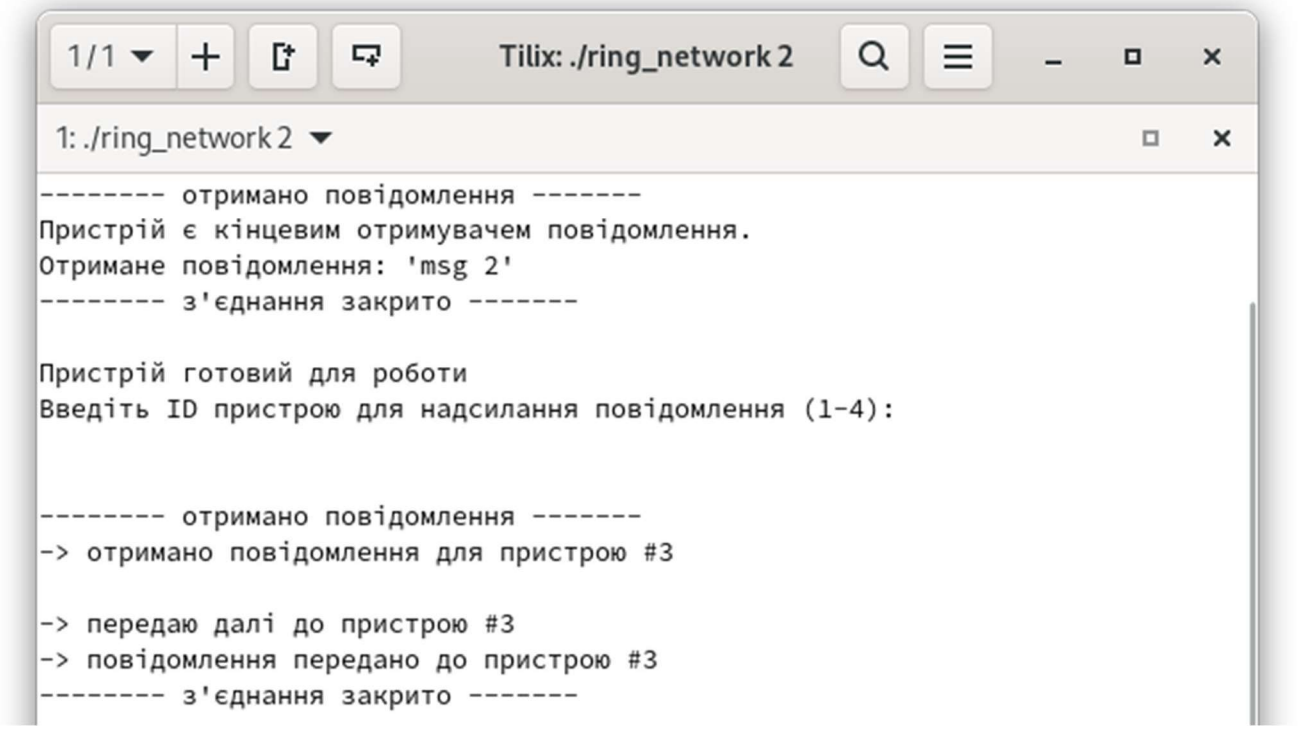


Рисунок 3.12 – Обробка повідомлення на пристрої №2

Після передачі повідомлення далі по пережі, повідомлення досягнуло цільового пристрою №3, як видно на рисунку 3.13. Оскільки повідомлення досягнуло цілі, його передача далі по мережі зупинилася.

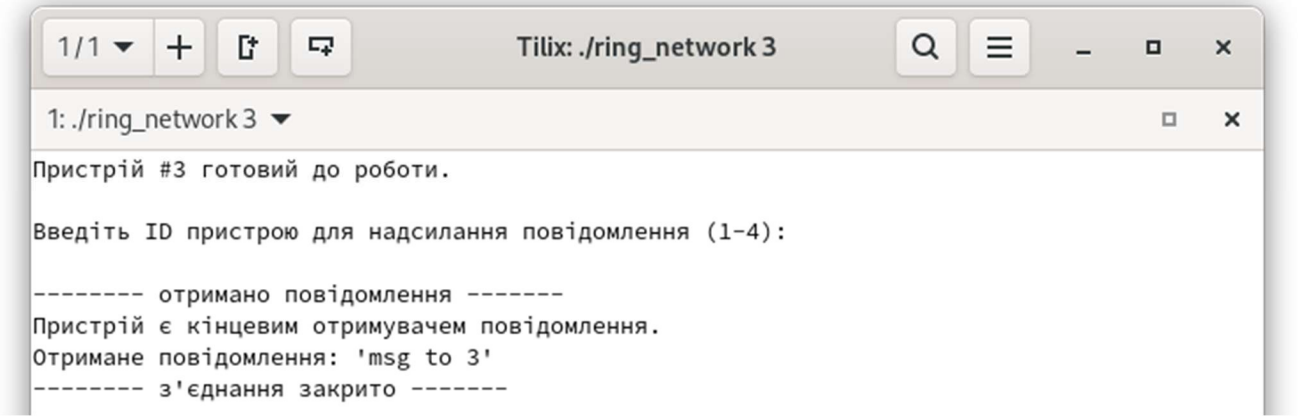


Рисунок 3.13 – Отримання повідомлення від серверу на пристрої №3

Тепер спробуємо передати повідомлення у зворотню сторону, тобто від пристрою №3 до серверу (рисунок 3.14)

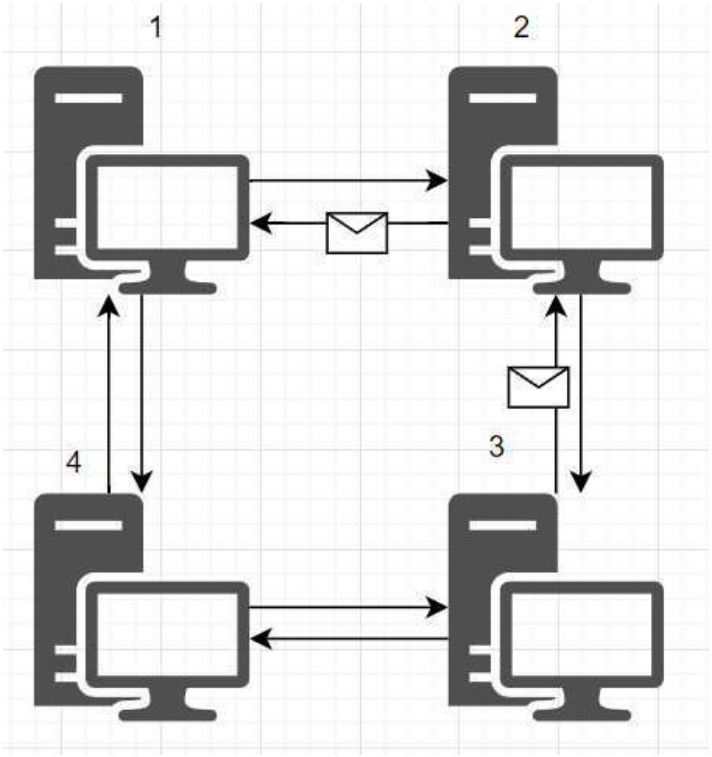


Рисунок 3.14 – Схема передачі повідомлення у зворотню сторону

Для цього на пристрої введемо номер 1 як отримувача та вкажемо вміст повідомлення (рисунок 3.15).

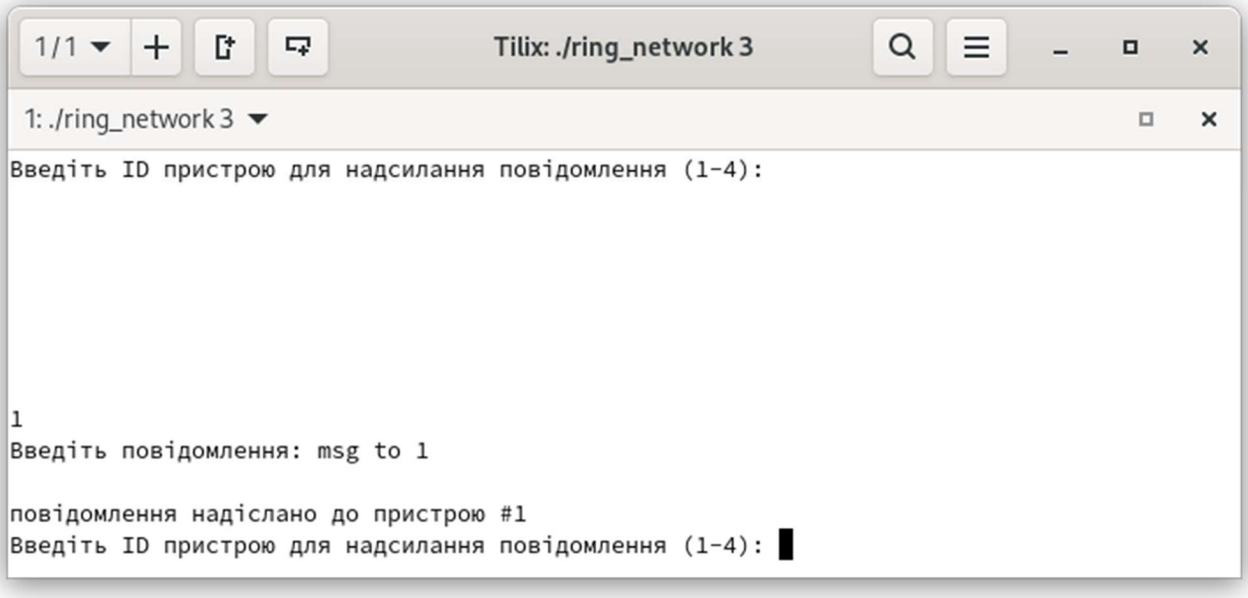


Рисунок 3.15 – Введення даних на пристрої №3

У даному випадку для надсилання алгоритм вибрав пристрій №2 для передачі повідмолення далі, що дозволено згідно двостороньої топології кільце. Проте також можна передати повідомлення через пристрій №4.

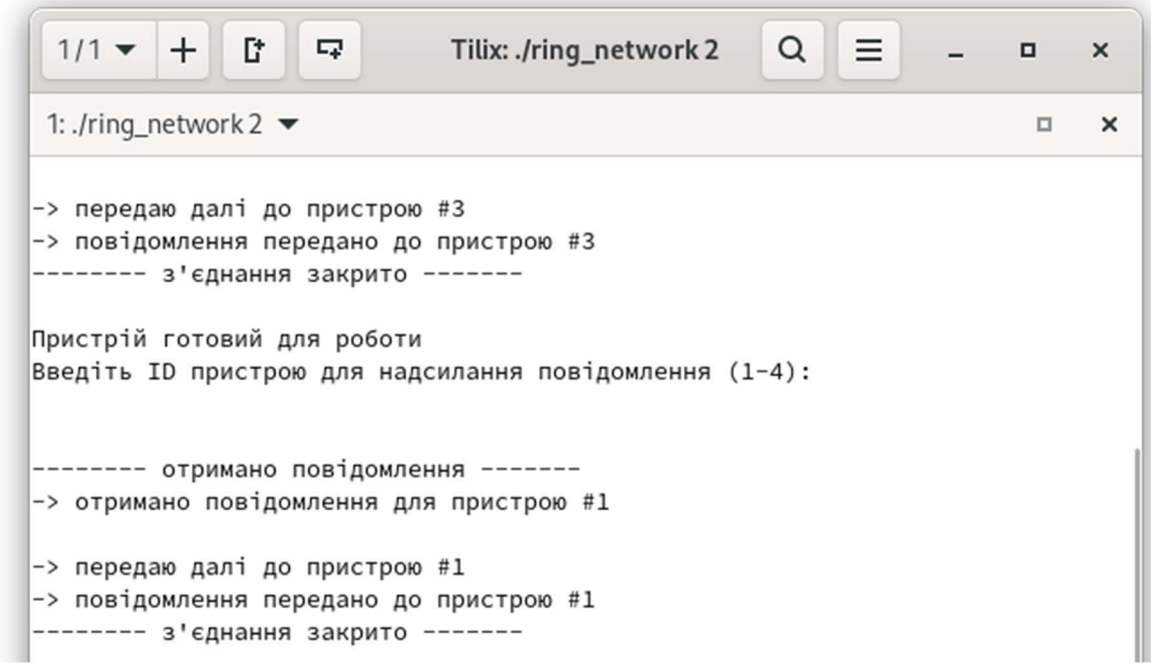


Рисунок 3.16 – Отримання повідомлення на пристрої №2

Після передачі по мережі повідомлення досягнуло серверу (рисунок 3.17)

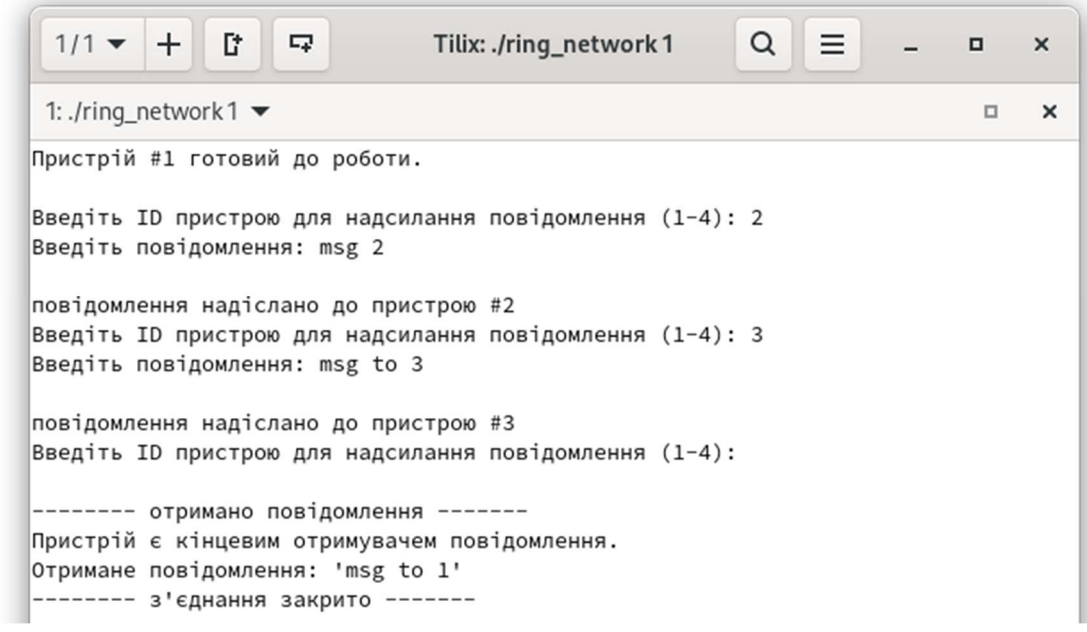


Рисунок 3.17 – Отримання повідомлення на сервері

3.5 Висновки

У третьому розділі було розроблено алгоритм для підключення пристроїв у мережі до сервера з метою кільцевої передачі повідомлень. Вибір мови C++ для написання проміжного програмного забезпечення зумовлений її перевагами для підключення мережевих пристроїв та компонентів, а також розробки кільцевих з’єднань у мультикомп'ютерних системах.

Особливості C++, такі як контроль над управлінням пам'яттю, підтримка багатопоточності та наявність потужних бібліотек, забезпечили високу продуктивність та стабільність системи.

Аналіз різних архітектур дозволив визначити, що клієнт-серверна архітектура найбільш підходить для забезпечення ефективної взаємодії компонентів системи. Ця архітектура сприяла швидкому та двонаправленому обміну даними між клієнтами та серверами.

Різні сценарії використання клієнт-серверної архітектури також були детально розглянуті. Приклади взаємодії між клієнтами та серверами, а також переваги та недоліки двонаправленого зв'язку, включно з використанням протоколу WebSocket для підвищення ефективності обміну даними, були детально описані.

В результаті перевірки програми, оскільки було обрано архітектуру кільце із двонаправленим зв’язком, алгоритм було перевірено на можливість надсилання повідомлень у дві сторони, тобто дозволив надіслати повідомлення у зворотню сторону, що відповідає вимогам до обраної архітектури.

Таким чином, третій розділ висвітлив ключові аспекти розробки та впровадження алгоритму підключення пристроїв у мультикомп'ютерній системі, вибір інструментів для розробки проміжного програмного забезпечення та архітектурні рішення, які сприяли створенню ефективної та надійної системи.

ВИСНОВКИ

Електронні обчислювальні машини (ЕОМ) та обчислювальна техніка зараз всюди. Вони грають важливу роль в сучасному світі, допомагаючи в бізнесі, науці, медицині та інженерії. Ці машини збирають, обробляють, зберігають та передають інформацію, що робить роботу в цих галузях швидшою та ефективнішою.

Сьогодні ЕОМ та обчислювальна техніка стали невід'ємною частиною життя майже кожної сучасної людини та організації. Вони допомагають збирати та аналізувати дані, створювати програмне забезпечення, візуалізувати та моделювати різні процеси. Також вони забезпечують комунікацію та зв'язок між людьми та системами.

У роботі було розглянуто мультикомп'ютерні системи, їх види та класифікації. Це системи з багатьох комп'ютерів, що працюють разом. Вони дуже надійні та можуть масштабуватися, розподіляючи навантаження між комп'ютерами. Це означає, що система працює навіть якщо один з комп'ютерів виходить з ладу.

Метою було створення мультикомп'ютерної системи на основі топології «подвійне кільце». Детально проаналізовано різні архітектури та обрано ту, яка найкраще підходить. Топологія «подвійне кільце» є надійною, бо кожен вузол з'єднаний з двома іншими що забезпечує передачу інформації навіть якщо один із каналів перестане працювати.

У першому розділі досліджено різні обчислювальні системи та типи паралельної обробки. Це допомогло обрати найкращу архітектуру для нашої системи.

У другому розділі докладно аналізували топології «кільце» та її розширена версія «подвійне кільце». Виділено основні відмінності між ними та проведено порівняльний аналіз їхніх структур. Визначено ключові параметри, як-от діаметр і зв'язність, та показано їхню ефективність на практиці. Окрім цього, розглянуто різні архітектури мультикомп’ютерних систем, що працюють з паралельною

обробкою даних, включаючи векторно-конвеєрні системи та системи з розподіленою пам'яттю. Особливо увагу звернено на масивно-паралельні комп'ютери (MPP), оскільки більшість сучасних потужних комп'ютерів використовують цю архітектуру. Для реалізації мультикомп'ютерної системи за топологією "подвійне кільце" обрано відповідне апаратне забезпечення: два ноутбуки, два комутатори, два маршрутизатори, два мережеві адаптери та кабелі для з'єднання кілець. Також встановлено додаткове програмне забезпечення. У цьому розділі наведено детальний опис та характеристики вибраного обладнання, зокрема назви та типи ноутбуків, комутаторів, маршрутизаторів, мережевих адаптерів та кабелів.

Третій розділ охоплює розробку алгоритмів підключення компонентів та програмного забезпечення. Розроблено мережеві протоколи для ефективного обміну даними та спеціалізоване програмне забезпечення для керування системою. Використання сучасних технологій дозволило створити систему, яка може застосовуватися в науці, бізнесі та інших галузях.

Отже, результатом виконання роботи є створення мультикомп'ютерної системи за топологією «подвійне кільце». Вона ефективна, надійна та гнучка, що робить її корисною для багатьох користувачів. Подальший розвиток включає оптимізацію алгоритмів та нові методи для підвищення ефективності системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Mehrotra P. and Van Rosendale J. Programming distributed memory architectures in Advances in Languages and Compilers for Parallel Computing. *MIT Press*, 2020. 10-15 p.

2. Литвин О. С. Співак, С. М. Паралельні та розподілені обчислення. 2014. 137 с.

3. Довідник з розподілених системю. URL: https://www.geeksforgeeks.org/distributed-systems-tutorial/ (дата звернення 05.04.2024)

4. Архітектура паралельних обчислювальних систем. URL: https://uk.wikipedia.org/wiki/Архітектура_паралельних_обчислювальних_систем (дата звернення 05.04.2024)

5. Тарарака В.Д. Архітектура комп'ютерних систем: навч. посіб. Житомир: ЖДТУ, 2018. 383 с.

6. Глоба Л.С. Розробка інформаційних ресурсів та систем. 2013. 378с.

7. Впровадження багатопроцесорних та мультикомп'ютерних технологій URL: https://www.geeksforgeeks.org/introduction-of-multiprocessor-and-multicomputer/ (дата звернення 07.04.2024)

8. Персональні комп'ютери. URL: https://www.britannica.com/technology/personal-computer/Faster-smaller-and-more-powerful-PCs (дата звернення 07.04.2024)

9. Що таке робочі станції. URL: https://www.geeksforgeeks.org/what-is-a-workstation/ (дата звернення 07.04.2024).

10. Mourad R., Sinoquet C., Zhang N. L., Liu T., Leray P. A survey on latent tree models and applications. *Journal of Artificial Intelligence Research*. 2013. Vol. 47. P. 157–203.

11. Joseph Faisal Nusairat. Rust for the IoT.2020.615 с.

12. Ornelas P., Nape I., Forbes E. Nonlocal skyrmions as topologically stable quantum entangled states of light. *Nature Photonics*. 2024. T. 18, № 3. P. 123–130. DOI: 10.1038/s41566-023-01360-4

13. Wi-Fi та як він працює. URL: https://www.cisco.com/c/en/us/products/wireless/what-is-wifi.html (дата звернення 08.04.2024)

14. Різниця 5G та 4G. URL: https://www.ericsson.com/en/5g/5g-vs-4g (дата звернення 08.04.2024)

15. Програмне забезпечення та операційна система. URL: https://naurok.com.ua/programne-zabezpechennya-operaciyna-sistema-ta-interfeys-270054.html (дата звернення 08.04.2024)

16. Сеті П., Саранг С. Р. Систематичний огляд літератури про шлюзи IoT ScienceDirect. 2021. 543 с.

17. Що таке Docker. URL: https://www.docker.com/resources/what-container/ (дата звернення 08.04.2024)

18. Віртуальні машини. URL: https://www.techopedia.com/definition/4805/virtual-machine-vm (дата звернення 08.04.2024)

19. Захист даних у комп'ютерних системах. URL: https://donec-anna.blogspot.com/2017/10/blog-post_24.html (дата звернення 12.04.2024)

20. Хмарні технології. URL: https://gigacloud.ua/blog/navchannja/scho-take-hmarni-tehnologii (дата звернення 12.04.2024)

21. Wicaksono A. B., Munadi R. Cloud server design for heavy workload gaming computing with Google cloud platform. International Journal of Electrical & Computer Engineering. 2023. Vol. 13(2). P. 2197

22. Що таке технології. URL: https://edin.ua/shho-take-xmarni-texnologi%D1%97-i-navishho-voni-potribni/ (дата звернення 15.04.2024)

23. Обчислювальна хмара. URL: https://futurenow.com.ua/shho-take-hmarni-tehnologiyi-ta-yak-tse-pratsyuye/ (дата звернення 15.04.2024)

24. Квантові технології. URL: https://tyzhden.ua/shcho-mozhut-kvantovi-kompiutery/ (дата звернення 15.04.2024)

25. Технології Token Ring і FDDI. URL: https://evileg.com/uk/post/43 (дата звереннення 16.04.2024)

26. Rolf Riesen. Operating System for Supercomputers. 2019. 540 c

27. Yadin A. Computer Systems Architecture. Chapman and Hall/CRC. 2016. 467 p.

28. Du D., Xu S., Cocquempot V. Observer-Based Fault Diagnosis and Fault-Tolerant Control for Switched Systems. Series: Studies in Systems, Decision and Control. Springer: Singapore. 2021. Vol. 280. 81 p.

29. Мережа FDDI. URL: https://ua5.org/lan/139-merezha-fddi.html (дата звернення 16.04.2024)

30. Характеристики технології FDDI. Особливості методу доступу в fddi. URL: https://studfile.net/preview/5797716/page:48/ (дата звернення 16.04.2024)

31. Протоколи передачі даних. URL: https://deltahost.ua/ua/tipi-merezhevix-protokoliv-i-ih-priznachennya-http-ip-ssh-ftp-pop3-mac.html (дата звернення 20.04.2024)

32. Системи з розподіленою пам'яттю. URL: https://studfile.net/preview/7396425/page:8/ (дата звернення 21.04.2024)

33. Топології комп'ютерних мереж. URL: https://marytisna.blogspot.com/2017/10/blog-post.html (дата звернення 21.04.2024)

34. Вступ до топології мережі. URL: https://www.cablify.ca/an-introduction-to-network-topology/ (дата звернення 22.04.2024)

35. Топологія кільце. URL: https://ajax.systems/ua/blog/advanced-security-ring-topology/ (дата звернення 22.04.2024)

36. Мережа Token-ring. URL: https://ua5.org/lan/137-merezha-token-ring.html (дата звернення 22.04.2024)

37. Комутатори та їх особливості. URL: https://stack-systems.com.ua/uk/blogs/shtcho-take-komutator-l3-rivnia-ta-v-chomu-joho-osoblyvosti (дата звернення 23.04.2024)

38. Мережеві карти. URL: https://artline.ua/uk/news/chto-takoe-setevaya-karta-i-kak-ee-vybrat (дата звернення 23.04.2024)

39. Сервер Dell PowerEdge T340. URL: https://www.dell.com/en-uk/shop/dell-poweredge-servers/poweredge-t340-tower-server/spd/poweredge-t340/pet3402a (дата звернення 23.04.2024)

40. Комп'ютер HP ProDesk 600 G6 Microtower. URL: https://chipchip.ua/product/hp-prodesk-600-g3-ip-00004464 (дата звернення 23.04.2024)

41. Комутатор Cisco Catalyst 2960 Series Switch. URL: https://www.cisco.com/c/en/us/support/switches/catalyst-2960-series-switches/series.html (дата звернення 23.04.2024)

42. Cisco ISR 4000 Series Integrated Services Router (марщрутизатор). URL: https://www.cisco.com/c/en/us/products/routers/4000-series-integrated-services-routers-isr/index.html (дата звернення 23.04.2024)

43. Tripp Lite series Duplex Multimode 50/125 Fiber Patch Cable. URL: https://tripplite.eaton.com/multimode-fiber-patch-cable-50-125-lc-lc-12m-40-ft~N52012M (дата звернення 23.04.2024)

44. Круцкевич Н. Перспективи розвитку коп'ютерних мереж з реконфігурацією архітектури на базі пам'яті колективного доступу. 2004.

45. Види архітектр. URL: https://www.esds.co.in/blog/cluster-computing-definition-architecture-and-algorithms/ (дата звернення 25.04.2024)

46. Jonathan M. D. Hill. An introduction to the data-parallel paradigm. *Department of Computer Science*, 1994. 699 – 714 p.

47. Ковтун, Л. О., Франчук, Р., Ткачук, В. М. Вибір архітектури програмного забезпечення інформаційної навчальної системи. 2019. 552 с.

48. Кластерна архітектура. URL: https://www.esds.co.in/blog/cluster-computing-definition-architecture-and-algorithms/ (дата звернення 26.04.2024дата)

49. Тарнавський Ю. А., Кузьменко І. М. Організація комп'ютерних мереж. 2018. 259 с.

50. Схема масивно паралельної архітектури. URL: https://www.linkedin.com/pulse/what-massively-parallel-processing-mpp-sankha-mitra (дата звернення 26.04.2024)

51. Грицюк Ю.І., Рак Т.Є. Програмування мовою С++: навчальний посібник. – Львів: Вид-во Львівського ДУ БЖД, 2011. 292 с. Статистика: іл. 10, табл. 18, бібліогр. 31.

52. Bertsekas Dimitri P. Parallel and Distributed Computation. Numerical Methods / Dimitri P. Bertsekas, John N. Tsitsiklis. – New Jersey: Prentice Hall, Englewood Cliffs, 1989. 718 p.

53. Socket парадигма програмування для зв'язку між пристроями. URL: https://polaridad.es/uk/socket-el-paradigma-de-programacion-para-comunicacion-entre-dispositivos/ (дата звернення 1.05.2024)

54. Тютюнник М. Паралельні алгоритми та засоби для розв'язання деяких задач масових обчислень. Конференція молодих вчених «Підстригачівські читання–20010». м. Львів, 25-26.

55. Ian Foster. Designing and Building Parallel Programs. Addison-Wesley, 1995. 370 p.

56. Лопандя, М. В. Обчислювальні системи з розподіленою пам'яттю. PhD Thesis. Сумський державний університет. 2013.

57. Русанова О. В., Писарчук О. О. Планування обчислень в паралельних та розподілених комп'ютерних системах. Лабораторний практикум. 2022.

58. Engelen, R. A. V. (2008). A framework for service-oriented computing with C and C++ Web service components. *ACM Transactions on Internet Technology (TOIT)*, 8(3), 1-25 p.

59. Moser, Louise E., and P. M. Melliar-Smith. Service-Oriented Architecture and Web Services. *Wiley Encyclopedia of Computer Science and Engineering*. 2007. №c93. 1-8 p.

60. Lin, M., Chen, Y., Hu, W. Middleware for Adaptive Data Transmission inSmart Grid Networks. *IEEE Transactions on Smart Grid*. 2022. №54. 5-12 p. DOI:10.1109/TSG.2022.3165678

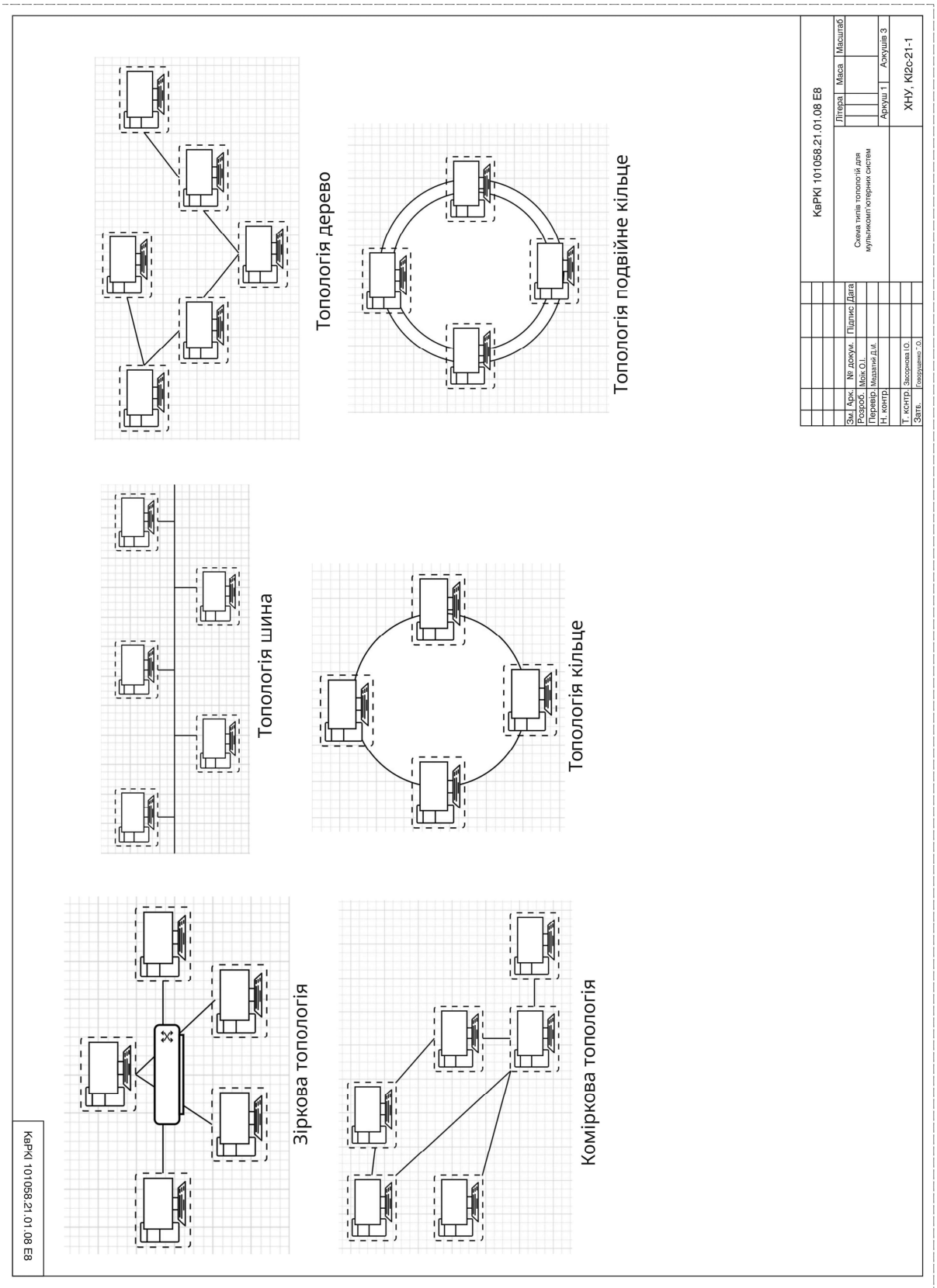
61. Kunduru A. R. Blockchain Technology for ERP Systems: A Review. *American Journal of Engineering, Mechanics and Architecture*. 2023. Vol. 1(7). P. 56- 63.

62. Aparecida Silva Camacho T., Martins do Rosario V., Oliveira Napoli O., Borin E. PB3Opt: Profile-based biased Bayesian optimization to select computing clusters on the cloud. *Concurrency and Computation: Practice and Experience* e7540. 2023. Vol. 35(18). P. 777-787.

Додаток А

(обов'язковий)

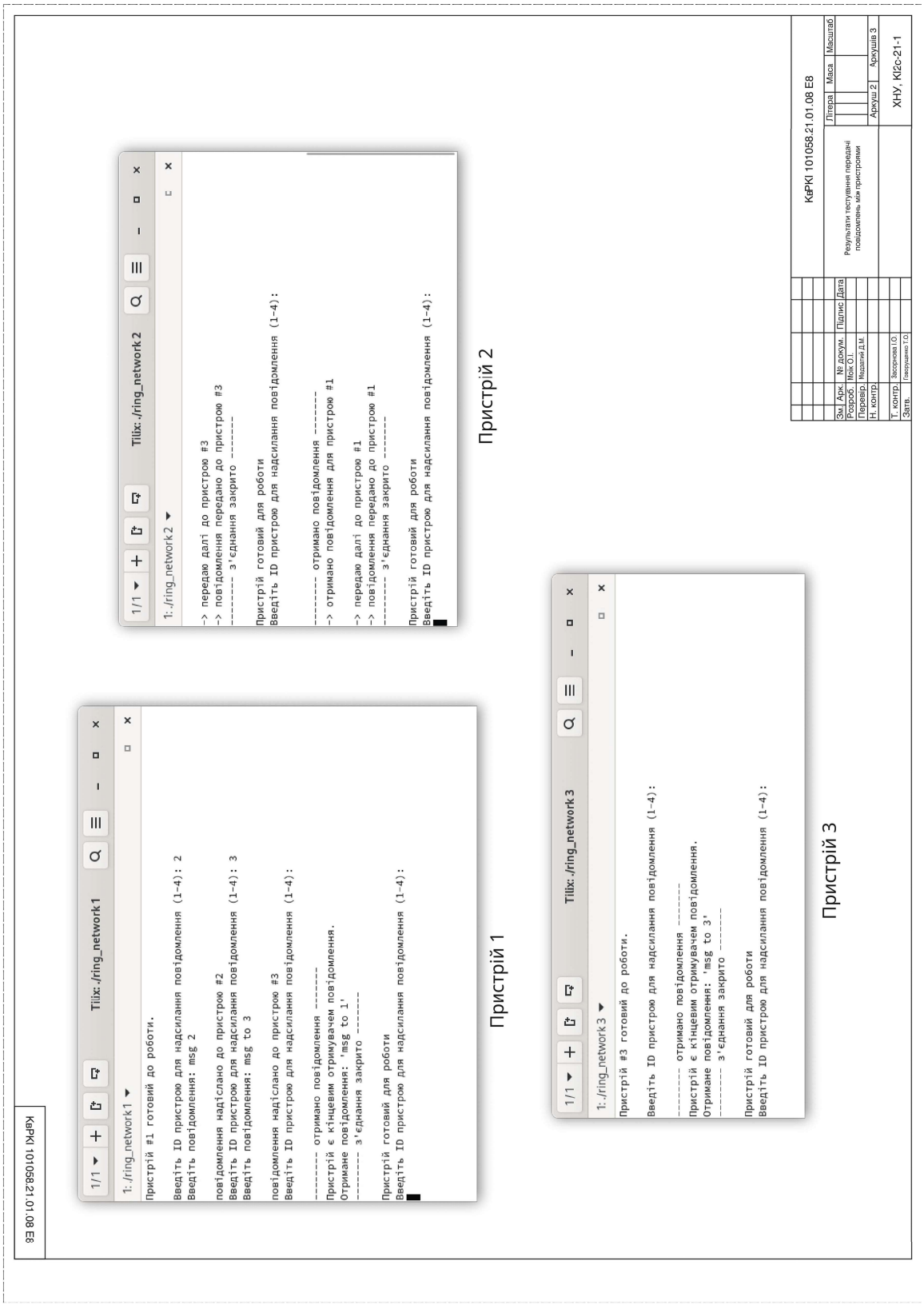
Копія креслення «Схема типів топологій для мультикомп'ютерних систем»



Додаток Б

(обов'язковий)

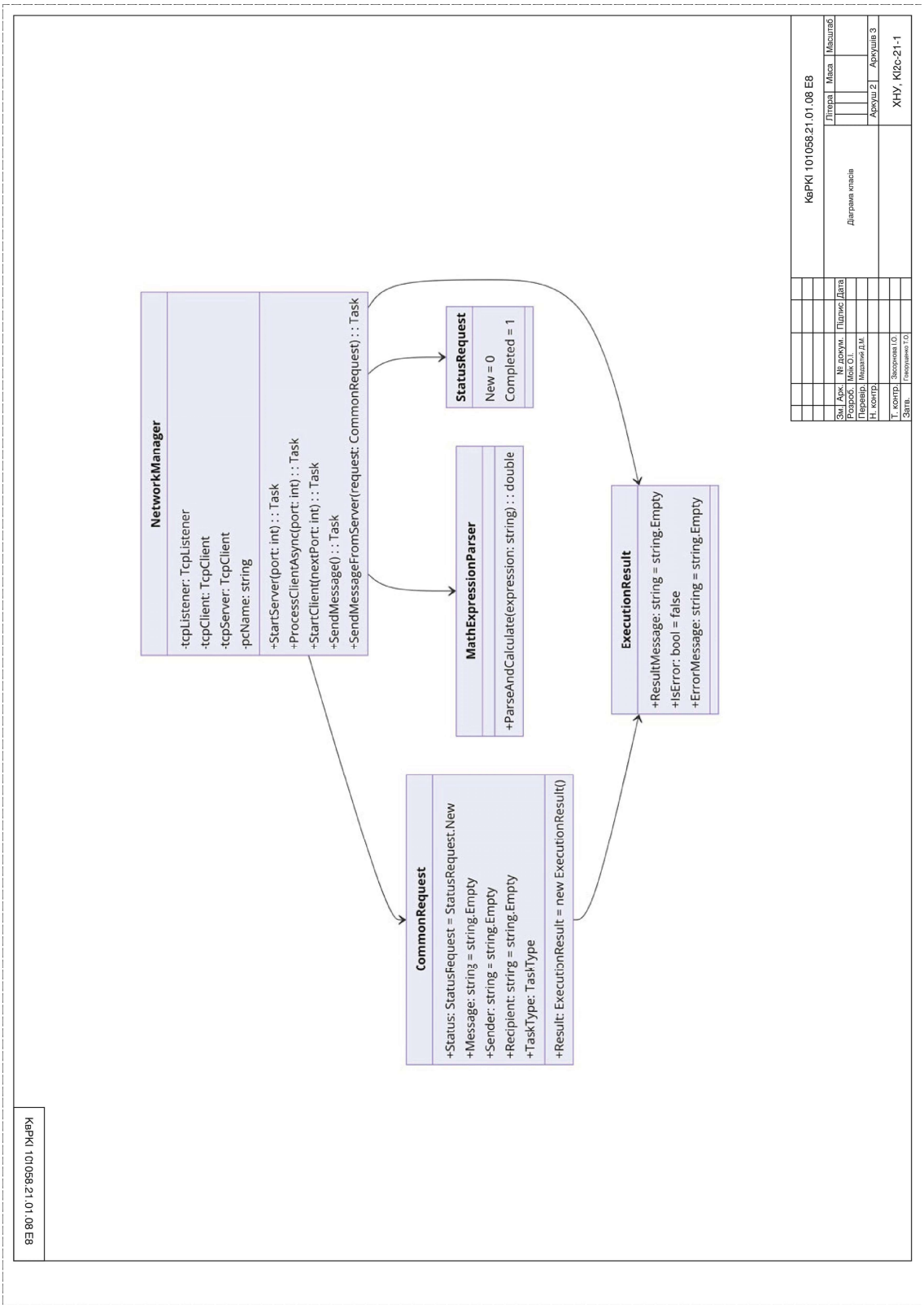
Копія креслення «Результати тестування передачі повідомлень між пристроями»



Додаток В

(обов'язковий)

Копія креслення «Діаграма класів»



Додаток Г

Лістинг коду проміжного програмного забезпечення

```
#include <iostream>
#include <thread>
#include <vector>
#include <string>
#include <cstring>
#include <arpa/inet.h>
#include <unistd.h>
#include <map>
#include <csignal>

using namespace std;

// Константи
const int BASE_PORT = 12300; // Базовий порт
const int NUM_DEVICES = 4; // Кількість пристроїв у кільці
const int MAX_RETRIES = 3; // Максимальна кількість спроб підключення

//  Обробка переривань
volatile bool running = true;

void signalHandler(int signum) {
    cout << "\nПереривання отримано (signal " << signum << "). Завершення роботи.\n";
    running = false;
}

// Структура для зберігання інформації про пристрій
struct Device {
    int id;
    int port;
};

// Структура для повідомлення
struct Message {
    int target_id;
    string text;
};

// todo
Device getNextDevice(int current_device_id, Message &msg, const map<int, Device> &devices) {
    int next_device_id = current_device_id % NUM_DEVICES + 1;

    /*
     *   | 1 | 2 | 3 | 4 |
     *---|---|---|---|---|
     * 1 | 0 | 1 | 0 | 1 |
     * 2 | 1 | 0 | 1 | 0 |
     * 3 | 0 | 1 | 0 | 1 |
```

```
     * 4 | 1 | 0 | 1 | 0 |
     *
     * */
    // cout << "\nd: using msg.target_id " << msg.target_id;
    switch (current_device_id) {
        case 1:
            // cout << "\nd: case 1";
            next_device_id = msg.target_id == 3 ? 2 : msg.target_id;
            break;
        case 2:
            // cout << "\nd: case 2";
            next_device_id = msg.target_id == 4 ? 1 : msg.target_id;
            break;
        case 3:
            // cout << "\nd: case 3";
            next_device_id = msg.target_id == 1 ? 2 : msg.target_id;
            break;
        case 4:
            // cout << "\nd: case 4";
            next_device_id = msg.target_id == 2 ? 3 : msg.target_id;
            break;
        default:
            break;
    }
    // cout << "\nd: next_device_id " << next_device_id;

    Device next_device = devices.at(next_device_id);
    // cout << "\nd: next_device " << next_device.id << " port: " << next_device.port;
    cout << "\n";

    return next_device;
}

// Функція для обробки отриманих повідомлень
void receiveMessages(int client_sock, int current_device_id, const map<int, Device> &devices) {
    char buffer[1024];
    while (true) {
        memset(buffer, 0, sizeof(buffer));
        int bytes_received = recv(client_sock, buffer, sizeof(buffer), 0);

        if (bytes_received > 0) {
            cout << "\n\n-------- отримано повідомлення -------\n";

            Message msg;
            memcpy(&msg.target_id, buffer, sizeof(msg.target_id));
            msg.text = string(buffer + sizeof(msg.target_id));

            if (msg.target_id == current_device_id) {
                cout << "Пристрій є кінцевим отримувачем повідомлення.\n";

                cout << "Отримане повідомлення: '" << msg.text << "'\n";
```

```
            break;
        } else {
            cout << "-> отримано повідомлення для пристрою #" << msg.target_id << "\n";

            // todo
            Device next_device = getNextDevice(current_device_id, msg, devices);

            cout << "-> передаю далі до пристрою #" << next_device.id << "\n";
            int forward_sock = socket(AF_INET, SOCK_STREAM, 0);
            if (forward_sock == -1) {
                cerr << "!!! Помилка створення клієнтського сокета: " << strerror(errno) << "\n";
                continue;
            }

            sockaddr_in server_addr;
            server_addr.sin_family = AF_INET;
            server_addr.sin_port = htons(next_device.port);
            inet_pton(AF_INET, "127.0.0.1", &server_addr.sin_addr);

            if (connect(forward_sock, (sockaddr *) &server_addr, sizeof(server_addr)) == -1) {
                cerr << "!!! Помилка з'єднання з пристроєм " << next_device.id << ": " << strerror(errno)
<< "\n";
                close(forward_sock);
                continue;
            }

            send(forward_sock, buffer, bytes_received, 0);
            close(forward_sock);
            cout << "-> повідомлення передано до пристрою #" << next_device.id << "\n";
        }
    } else if (bytes_received == 0) {
        break;
    } else {
        cerr << "!!!  Помилка отримання даних: " << strerror(errno) << "\n";
        break;
    }
  }
  close(client_sock);

  cout << "-------- з'єднання закрито -------\n";

  cout << "\nПристрій готовий для роботи\nВведіть ID пристрою для надсилання повідомлення
(1-" << NUM_DEVICES << "):\n";
  cin.clear();
}

// Функція для відправлення повідомлень
void sendMessages(int sock, int device_id, const Message &msg) {
  char buffer[1024];
  memcpy(buffer, &msg.target_id, sizeof(msg.target_id));
  memcpy(buffer + sizeof(msg.target_id), msg.text.c_str(), msg.text.size() + 1);
```

```
    int bytes_sent = send(sock, buffer, sizeof(msg.target_id) + msg.text.size() + 1, 0);
    if (bytes_sent != -1) {
        cout << "повідомлення надіслано до пристрою #" << msg.target_id << endl;
    } else {
        cerr << "!!! Помилка надсилання даних: " << strerror(errno) << "\n";
    }
}

// Функція для обробки вводу користувача та надсилання повідомлень
void handleUserInput(int current_device_id, const map<int, Device> &devices) {
    while (true) {
        sleep(1);

        int target_id;
        string text;

        cout << "Введіть ID пристрою для надсилання повідомлення (1-" << NUM_DEVICES << "):
";
        cin >> target_id;
        cin.ignore();
        cin.clear();

        if (target_id < 1 || target_id > NUM_DEVICES) {
            cerr << "!!! Неправильний ID пристрою: " << target_id << endl;
            continue;
        }

        cout << "Введіть повідомлення: ";
        getline(cin, text);

        Message msg = {target_id, text};
        Device next_device = getNextDevice(current_device_id, msg, devices);

        int client_sock = socket(AF_INET, SOCK_STREAM, 0);
        if (client_sock == -1) {
            cerr << "!!! Помилка створення клієнтського сокета: " << strerror(errno) << "\n";
            continue;
        }

        sockaddr_in server_addr;
        server_addr.sin_family = AF_INET;
        server_addr.sin_port = htons(next_device.port);
        inet_pton(AF_INET, "127.0.0.1", &server_addr.sin_addr);

        int retries = 0;
        while (connect(client_sock, (sockaddr *) &server_addr, sizeof(server_addr)) == -1 && retries <
MAX_RETRIES) {
            cerr << "!!! Помилка з'єднання з пристроєм #" << next_device.id << ": " << strerror(errno) <<
". Спроба "
                 << retries + 1 << " з " << MAX_RETRIES << "\n";
            retries++;
```

```
            this_thread::sleep_for(chrono::seconds(1));
        }

        if (retries == MAX_RETRIES) {
            cerr << "!!! Не вдалося з'єднатися з пристроєм #" << next_device.id << " після " <<
MAX_RETRIES
                 << " спроб.\n";
            close(client_sock);
            continue;
        }

        sendMessages(client_sock, current_device_id, msg);
        close(client_sock);
    }
}

int main(int argc, char *argv[]) {
    // register signal
    // signal(SIGINT, signalHandler);
    system("clear");

    map<int, Device> devices = {};
    for (int id = 1; id <= NUM_DEVICES; ++id) {
        devices[id] = {id, BASE_PORT + id};
        // cout << "Registerd: id:" << devices[id].id << " port:" << devices[id].port << "\n";
    }

    if (argc != 2) {
        cerr << "!!! Використання: " << argv[0] << " <current_device_id>\n";
        return 1;
    }

    int current_device_id = stoi(argv[1]);

    if (current_device_id < 1 || current_device_id > NUM_DEVICES) {
        cerr << "!!! Вказано неправильний ID пристрою.\n";
        return 1;
    }

    Device &device = devices[current_device_id];

    // Створення сокета
    int sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock == -1) {
        cerr << "!!! Помилка створення сокета: " << strerror(errno) << "\n";
        return 1;
    }

    sockaddr_in addr;
    addr.sin_family = AF_INET;
    addr.sin_port = htons(device.port);
```

```
    addr.sin_addr.s_addr = INADDR_ANY;

    // Прив'язка сокета
    if (bind(sock, (sockaddr *) &addr, sizeof(addr)) == -1) {
        cerr << "!!! Помилка прив'язки сокета: " << strerror(errno) << "\n";
        close(sock);
        return 1;
    }

    // Слухання з'єднань
    if (listen(sock, NUM_DEVICES) == -1) {
        cerr << "!!! Помилка встановлення режиму прослуховування: " << strerror(errno) << "\n";
        close(sock);
        return 1;
    }

    cout << "Пристрій #" << current_device_id << " готовий до роботи.\n\n";

    // Прийняття вхідних з'єднань у окремому потоці
    thread receiver([&]() {
        while (running) {
            int client_sock;
            sockaddr_in client_addr;
            socklen_t client_size = sizeof(client_addr);

            client_sock = accept(sock, (sockaddr *) &client_addr, &client_size);
            if (client_sock == -1) {
                if (errno == EINTR) break; // Перевірка на переривання
                cerr << "!!! Помилка прийняття з'єднання: " << strerror(errno) << "\n";
                continue;
            }

            thread(receiveMessages, client_sock, current_device_id, devices).detach();
        }
    });

    // Створення окремого потоку для обробки вводу користувача
    thread userInputThread(handleUserInput, current_device_id, devices);

    while (running) {
        this_thread::sleep_for(chrono::milliseconds(100));
    }

    receiver.join();
    userInputThread.join();
    close(sock);

    return 0;
}
```

UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1016383982

Дата перевірки:
23.06.2024 19:38:17 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
23.06.2024 20:34:27 EEST

ID користувача:
100005591

Назва документа: Моїк_Проміжне програмне забезпечення мультикомп'ютерної системи з топологією "кільц...

Кількість сторінок: 73 Кількість слів: 11050 Кількість символів: 89286 Розмір файлу: 2.69 MB ID файлу: 1016194555

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

13.9%
Схожість

Найбільша схожість: 8.67% з джерелом з Бібліотеки (ID файлу: 1016124915)

4.69% Джерела з Інтернету	172	Сторінка 75
11.1% Джерела з Бібліотеки	173	Сторінка 77

0.26% Цитат

Цитати	1	Сторінка 78
Посилання	1	Сторінка 78

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	35
Підозріле форматування	12 сторінок

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 6.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 10%**

ID: 132294 Назва: БКР Проміжне програмне забезпечення мультикомп'ютерної системи з топологією "кільце" двонаправленого зв'язку Додано в БД: 2024-06-23 Автора: О.І. Моїк Керівники: Д.М. Медзатий Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	78294	646	6923 (9%)	75 (12%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Моїк Олександр Ігорович

Тема: Проміжне програмне забезпечення мультикомп'ютерної системи з топологією "кільце" двонаправленого зв'язку

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 Кількість сторінок записки 63

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є створення програмного забезпечення для мультикомп'ютерної системи з топологією "кільце" двонаправленого зв'язку.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі кваліфікаційної роботи проведено дослідження предметної області (проаналізовано існуючі рішення та проведено порівняльний аналіз) та виконано постановку задачі дослідження. В другому розділі кваліфікаційної роботи проведено аналіз та вибір мови програмування та апаратних пристроїв. В третьому розділі кваліфікаційної роботи виконано апаратну та програмну реалізацію мультикомп'ютерної системи типу кільце двонаправленого типу, а саме реалізовано підключення кінцевих пристроїв та розроблено програмне забезпечення для робити з ними.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: відсутність графічного інтерфейсу для програми.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Бодрашов Леонід Петрович, зав. кафедри ІПЗ ХНУ

"24" червня 2024 р. _____(підпис)

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Моїка Олександра Ігоровича
ПІБ здобувача вищої освіти

ФІТ, 3 курсу, групи КІ2с-21-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений(а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

22 квітня 2024 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Проміжне програмне забезпечення мультикомп'ютерної системи з топологією "кільце" двонаправленого зв'язку

Автор: Моїк Олександр Ігорович

Спеціальність: 123 – Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Медзатий Дмитро Миколайович, д.т.н., доцет

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ідентичності/схожості, складає 13.9% і адресується до 344 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Д.М. Мездатий

Гарант ОП _____ С.М. Лисенко

Завідувач кафедри КІПС _____ Т.О. Говорущенко