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

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Мультикомп'ютерна система згідно сотової топології
Назва теми

КвРКІ.190126.19.01.12 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 IV курсу, група КІ2–19–1 _____ Д.А. Крижанівський
Підпис Ініціали, прізвище

Керівник _____ К.М. Березька
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ С.М. Лисенко
Підпис, дата Ініціали, прізвище

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

«5» червня 2023 р.

Хмельницький 2023

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

"11" 01 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Крижанівському Дмитру Андрійовичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Мультикомп'ютерна система згідно сотової топології

Керівник проекту (роботи) Березька К. М., к.т.н., доцент

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

Затверджена наказом ректора університету від 1.03.2023 р. № 5

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2023 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

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

Дослідження предметної області та постановка задачі

Архітектура апаратного забезпечення комп'ютерної системи

Архітектура програмного забезпечення комп'ютерної системи

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

Схеми топологій для мультикомп'ютерних систем

UML-діаграма програмного забезпечення та під'єднання складових системи

Схеми централізованих, децентралізованих та розподілених систем

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., професор кафедри КПС		
Антиплагіат	Нічепорук А.О., доцент кафедри КПС		

7. Дата видачі завдання « 01 » 03 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	20.02.2022	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.03.2023	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	10.03.2023	виконано
4	Робота над розділом 2 – архітектура апаратного забезпечення комп'ютерної системи	20.04.2023	виконано
5	Робота над розділом 3 – архітектура програмного забезпечення комп'ютерної системи	30.04.2023	виконано
6	Оформлення пояснювальної записки згідно вимог	31.05.2023	виконано
7	Попередній захист ВКР	02.06.2023	виконано
8	Захист ВКР на засіданні ЕК	Червень 2023 року	

Студент _____ Підпис Д. А. Крижанівський Ініціали, прізвище

Керівник проекту (роботи) _____ Підпис К. М. Березька Ініціали, прізвище

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ.190126.19.01.12 ПЗ	Пояснювальна записка	55		
			Графічні матеріали			
2		КвРКІ.190126.19.01.12 Е8	Схеми топологій для	1		
			мультикомп'ютерних			
			систем			
3		КвРКІ.190126.19.01.12 Е8	UML-діаграма	1		
			програмного забезпечення			
			та під'єднання складових			
			системи			
4		КвРКІ.190126.19.01.12 Е8	Схема централізованих,	1		
			децентралізованих та			
			розподілених систем			

Зм	Арк	№ докум	Підпис	Дата	КвРКІ.190126.19.01.12 ПЗ		
Розробив		Кожанівський			Відомість проекту	Літера	Аркуш	Аркушів
Перевір.		Березька		05.06		У	1	1
Н. контр.		Лисенко				ХНУ, КІ2-19-1		
Затв.		Говорущенко		05.06				

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Мультикомп'ютерна система згідно сотової топології».

Автор роботи: Крижанівський Дмитро Андрійович.

Керівник роботи: Березька Катерина Миколаївна.

Пояснювальна записка: 55 с., 30 рис., 4 дод., 60 джерел.

Графічна частина: 15 презентаційних слайдів.

МУЛЬТИКОМП'ЮТЕРНА СИСТЕМА, ТОПОЛОГІЯ, ПРОГРАМУВАННЯ, C++.

Метою роботи є проектування та моделювання мультикомп'ютерної системи згідно сотової топології на мові C++.

У цій роботі спроектована та змодельована мультикомп'ютерна система згідно сотової топології на мові C++. Для розробки системи було вирішено використати 3 ноутбуки, 3 точки доступу та антени для правильної роботи точок доступу. Було розроблене проміжне програмне забезпечення на мові C++, що дозволяє обмінюватись даними між ноутбуками та точками доступу.

Підпис студента _____ Дата 05.06.2023

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ЕОМ – Електронна обчислювальна машина

МКС – Мультикомп'ютерна система

ОС – Операційна система

REST – Representational State Transfer (передача репрезентативного стану)

JSON – JavaScript Object Notation (запис об'єктів JavaScript)

XML – Extensible Markup Language (розширювана мова розмітки)

SOAP – Simple Object Access Protocol (протокол обміну структурованими повідомленнями)

HTTP – HyperText Transfer Protocol (протокол передачі гіпертекстових документів)

IP – Internet Protocol (протокол мережевого рівня)

API – Application programming interface (прикладний програмний інтерфейс)

LAN – Local area network (локальна комп'ютерна мережа)

FDMA – Frequency-division multiple access (множинний доступ з частотним поділом)

TDMA – Time division multiple access (метод часового поділу одного фізичного каналу зв'язку)

SSID – Service Set Identifier (унікальне найменування бездротової мережі)

DHCP – Dynamic Host Configuration Protocol (протокол динамічної конфігурації вузла)

DNS – Domain Name System (система доменних імен)

IDE – Integrated development environment (інтегроване середовище розробки)

UDP – User Datagram Protocol (протокол датаграм користувача)

ВСТУП

Електронні обчислювальні машини (ЕОМ) та обчислювальна техніка є технологіями, які забезпечують збір, обробку, зберігання та передачу інформації, і мають велике значення для різних галузей, таких як бізнес, наука, медицина та інженерія. З початку 20-го століття ЕОМ змінили світ, дозволивши виконувати складні обчислення та обробляти інформацію. Сьогодні ЕОМ та обчислювальна техніка стали невід'ємною частиною більшості сучасних організацій, використовуються для збору та аналізу даних, розробки програмного забезпечення, візуалізації та симуляції процесів, а також для забезпечення комунікації та зв'язку [1-2].

Мультикомп'ютерні системи - це комп'ютерні системи, що складаються з взаємопов'язаних комп'ютерів, які працюють як єдине ціле. Кожен комп'ютер зазвичай має свою власну пам'ять та процесор, але може обмінюватися даними та задачами через мережу. Мультикомп'ютерні системи мають високий рівень масштабованості, оскільки можуть бути розширені додаванням нових комп'ютерів, які будуть виконувати нові завдання. Це забезпечує велику гнучкість та можливість збільшення продуктивності системи в залежності від потреб користувачів. Крім того, мультикомп'ютерні системи забезпечують високу надійність, оскільки якщо один з комп'ютерів в системі відмовить, інші комп'ютери можуть продовжувати роботу, забезпечуючи безперервну роботу системи.

Ця робота має на меті створення універсальної мультикомп'ютерної системи з використанням сотової топології. Універсальна мультикомп'ютерна система з використанням сотової топології пропонує ряд переваг у порівнянні з іншими архітектурними рішеннями. Сотова топологія передбачає, що система складається з великої кількості взаємопов'язаних комп'ютерних вузлів, які утворюють соти або клітки. Кожна клітина має свою локальну пам'ять та процесор, що забезпечує високу рівень локальності даних і обчислень. Більше того, зв'язок між клітинами здійснюється через спеціальні комутатори, що дозволяє розподіляти завдання та обмінюватися даними ефективно. У рамках цієї роботи будуть проаналізовані різні архітектури, які можуть бути використані в системах з загальною пам'яттю, а також

буде проведений аналіз існуючих архітектурних рішень. У другому та третьому розділах буде описано процес проектування та реалізації цієї системи з детальним поясненням вибору та застосування апаратно-програмної бази.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Мультикомп'ютерні системи (МКС) - це комп'ютерні системи, які складаються з багатьох комп'ютерів, що працюють разом у відповідності до спільної задачі. У цих системах кожен комп'ютер виконує свою окрему роль, тобто вони можуть виконувати різні операції, обчислювати різні алгоритми, а потім передавати результати іншим комп'ютерам для обробки.

Мультикомп'ютерні системи широко використовуються у багатьох сферах, зокрема у високопродуктивному обчисленні, великих наукових проектах, обробці даних та багатокористувацьких системах. Вони забезпечують високу продуктивність та можливість обробки великого обсягу даних [3-4].

Одним з прикладів МКС є кластери комп'ютерів, які використовуються для високопродуктивного обчислення. У таких системах кожен комп'ютер виконує окремі операції, а потім передає результати іншим комп'ютерам для обробки. Це дозволяє забезпечити високу продуктивність та швидкість обробки даних.

Інший приклад МКС - це розподілені системи, які використовуються для обробки даних у великих корпоративних середовищах. У таких системах різні комп'ютери виконують різні функції, такі як зберігання даних, обробка транзакцій та моніторинг системи.

Мультикомп'ютерні системи можуть бути використані для вирішення дуже великих та складних задач, які вимагають великої обчислювальної потужності. Це можуть бути задачі моделювання клімату, обробки великих об'ємів даних, створення складних графічних ефектів у кіно та відеоіграх, або розрахунки складних наукових моделей.

Незважаючи на всі переваги мультикомп'ютерних систем, вони також мають свої недоліки. Один з них - складність управління та координації роботи різних

комп'ютерів в системі. Необхідно мати спеціальні знання та навички, щоб ефективно використовувати такі системи.

Крім того, мультикомп'ютерні системи можуть бути вразливі до атак з зовнішнього середовища, тому необхідно використовувати відповідні засоби захисту та безпеки. Також, додавання нових комп'ютерів до системи може призвести до складнощів з підтримкою та сумісністю програмного забезпечення [5].

Усі ці фактори потрібно враховувати при виборі технологій та обладнання для створення мультикомп'ютерних систем. Такі системи можуть бути дуже потужним та ефективним інструментом для вирішення складних задач та забезпечення високої продуктивності, але вони потребують відповідного підходу та управління.

Один з ключових аспектів аналізу МКС - це архітектура системи, яка визначає спосіб, яким комп'ютери взаємодіють і обмінюються даними. Різні архітектури можуть мати різні переваги і недоліки, залежно від конкретних вимог системи.

Другим важливим аспектом є програмне забезпечення МКС. У таких системах необхідно мати ефективні алгоритми розподілу завдань між комп'ютерами, а також ефективну систему керування ресурсами та забезпечення взаємодії між комп'ютерами. Розробка програмного забезпечення МКС є складною задачею, оскільки потрібно забезпечити ефективність і надійність системи, а також забезпечити безпеку даних [6].

Проблеми ефективності також є важливим аспектом аналізу МКС. При проектуванні таких систем потрібно враховувати розмір та складність задач, які будуть виконуватися, а також кількість комп'ютерів, які будуть використовуватися. Для досягнення максимальної ефективності необхідно розробляти ефективні алгоритми розподілу завдань між комп'ютерами та враховувати обмеження швидкості передачі даних між ними.

Надійність та безпека є також важливими аспектами аналізу МКС. Надійність роботи системи та захист від вторгнень можуть бути складними завданнями для МКС, оскільки вони можуть містити багато комп'ютерів, які взаємодіють між

собою через мережу. Для досягнення високого рівня надійності, системи повинні бути спроектовані з урахуванням можливих відмов комп'ютерів та мережевих з'єднань, і мати механізми відновлення роботи у разі непередбачуваних ситуацій.

Захист від вторгнень є іншим важливим аспектом аналізу МКС. Зокрема, важливо забезпечити захист від несанкціонованого доступу до даних та захист від зломів системи. Для досягнення цього необхідно використовувати ефективні методи аутентифікації та авторизації, шифрування даних та контролювати доступ до ресурсів системи.

МКС є важливою складовою сучасних комп'ютерних систем, оскільки вони дозволяють забезпечити високу продуктивність та обробку великого обсягу даних. При проектуванні таких систем необхідно враховувати потреби користувачів та використовувати ефективні алгоритми для оптимізації роботи системи

Узагальнюючи, аналіз МКС включає дослідження архітектури системи, програмного забезпечення, ефективності, надійності та безпеки. При проектуванні таких систем необхідно враховувати потреби користувачів та використовувати ефективні алгоритми для оптимізації роботи системи та забезпечення безпеки даних [7].

1.2 Аналіз наявного програмно-апаратного забезпечення

Універсальні комп'ютерні системи великого розміру, такі як мейнфрейми, є найпотужнішими комп'ютерами загального призначення, що забезпечують неперервний режим роботи протягом усього дня. Вони можуть містити один або кілька процесорів, кожен з яких може бути оснащений спеціалізованими векторними процесорами, що забезпечують продуктивність суперкомп'ютерів. Зазвичай мейнфрейми асоціюються з великими габаритами та вимагають спеціально обладнаних приміщень з водяним охолодженням і кондиціонуванням. Однак завдяки прогресу в елементній базі та конструкції, розміри основних пристроїв значно зменшилися. Поряд з надзвичайно потужними мейнфреймами, які потребують подвійної водяної системи охолодження, існують менш потужні моделі, для яких достатньо примусової повітряної вентиляції, а також модульні

моделі, які можуть працювати без необхідності спеціальних приміщень і кондиціонерів [8-9].

Мейнфрейми постачаються такими комп'ютерними компаніями, як IBM, Amdahl (у складі Fujitsu), ICL, Fujitsu Siemens Computers, Wincor Nixdorf AG, Siemens Business Services та іншими. Проте, ведучу роль без сумніву відіграє компанія IBM. Їх архітектура системи IBM/360, запущена в 1964 році, та її наступні покоління стали зразком для наслідування. Наприклад, в СРСР на протязі багатьох років були випущені машини серії ЄС ЕОМ, які були аналогами цієї системи. Мейнфрейми, з архітектурної точки зору, є багатопроцесорними системами, що включають один або кілька центральних та периферійних процесорів з загальною пам'яттю, з'єднаних швидкими магістралами передачі даних. При цьому основне обчислювальне навантаження покладено на центральні процесори, а периферійні процесори (у термінології IBM - селекторні, блок-мультиплексні, мультиплексні канали та процесори телеопрацювання) забезпечують взаємодію з різноманітними периферійними пристроями [10-11].

Спочатку мейнфрейми працювали в централізованій моделі обчислень та використовували патентовані операційні системи. Однак зростаючий інтерес споживачів до відкритих систем, заснованих на міжнародних стандартах, змусив постачальників мейнфреймів значно розширити можливості своїх операційних систем для досягнення сумісності.

Один з найпотужніших мейнфреймів на сьогоднішній день - це сім'я мейнфреймів zSeries від компанії IBM. Основні особливості цих комп'ютерів включають наступне: велика ємність зовнішньої пам'яті, що адресується за допомогою 64-розрядного адресного слова, розміщення до 54 центральних процесорів в кожному з 32 блоків, при тому, що кожен блок може бути розташованим на відстані до 100 кілометрів, забезпечуючи продуктивність понад 18 мільярдів команд за секунду, підтримка операційних систем Linux, z/OS, z/VM, z/VSE, z/TPF, MU-SIC/SP, потужні зовнішні інтерфейси [12].

Стрімкий зросток продуктивності персональних комп'ютерів, робочих станцій і серверів спричинив зменшення попиту на мейнфрейми. Однак останнім часом цей процес сповільнився. Експерти вважають, що основною причиною

відродження інтересу до мейнфреймів є складність переходу до розподіленої архітектури клієнт-сервер, яка виявилася складнішою, ніж очікувалося. Багато користувачів також вважають, що розподілене середовище не є належно надійним для найвідповідальніших застосувань, відмінно від мейнфреймів.

Нинішнім недоліком мейнфреймів є відносно низьке співвідношення продуктивність/вартість. Проте, постачальники мейнфреймів докладають значних зусиль для поліпшення цього показника. Варто також пам'ятати, що по всьому світу існує велика інсталяційна база мейнфреймів, на яких працюють десятки тисяч прикладних програмних систем. Відмовитися від розробленого протягом років програмного забезпечення є просто нерозумним. Тому на даний момент очікується зростання обсягів продажу мейнфреймів, які, з одного боку, дозволять модернізувати існуючі системи з скороченням експлуатаційних витрат, а з іншого боку, створять нову базу для найвідповідальніших застосувань [13].

1.3 Аналіз системного програмного забезпечення

Операційна система (ОС) є програмним забезпеченням, яке дозволяє ефективно використовувати комп'ютерне обладнання зручним для користувача способом. Вона складається з набору програмних модулів, що забезпечують користувачеві керування комп'ютером та взаємодію програм з зовнішніми пристроями та один з одним. ОС є ключовою складовою системного програмного забезпечення і працює на основі команд. Операційні системи можна класифікувати за наступними критеріями:

- локальні операційні системи;
- мережні операційні системи;
- операційні системи розподілених систем (Cloud Computing);
- спеціалізовані операційні системи;
- операційні системи реального часу.

ОС також можна класифікувати з точки зору їх можливостей:

- універсальні (призначені для загального використання), спеціальні (призначені для вирішення конкретних задач) та спеціалізовані (працюють на спеціальному обладнанні);
- підтримка одно- або багатозадачності;
- підтримка багатокористувацького режиму;
- реалізація багатозадачності з примусовим або без примусового переключення;
- підтримка багатоядерності/багатопроцесорності;
- вбудовані або невбудовані ОС;
- відкриті або закриті за розширенням;
- вільний або комерційний доступ до вихідного коду.

ОС може розглядатися як розширення апаратної частини комп'ютера, що приховує роботу заліза, або як система управління ресурсами, яка здійснює планування та використання ресурсів, а також моніторинг поточного стану використання ресурсів [14-15].

Основні функції операційних систем включають:

- планування завдань та використання процесора, включаючи обробку переривань і забезпечення багатозадачної роботи;
- забезпечення комунікації та синхронізації процесів;
- забезпечення інтерфейсу між прикладними програмами та службовими сервісами ОС;
- управління пам'яттю;
- управління файловою системою;
- управління введенням/виведенням;
- забезпечення безпеки [16].

Операційна система складається з таких компонентів:

- ядро, яке є центральною частиною ОС і забезпечує координований доступ прикладних програм до ресурсів комп'ютера, таких як процесорний час, оперативна пам'ять, зовнішні пристрої введення та виведення інформації. Воно перекладає команди з мови прикладних програм в мову двійкового коду, яку розуміє комп'ютер;

- драйвери, які є програмами для перекладу вказівок комп'ютера мовою певного пристрою, такого як принтер, сканер, звукова або відеокарта, і навпаки;
- утиліти, які є допоміжними програмами, що обслуговують диски, перевіряють комп'ютер, налаштовують параметри роботи;
- інтерфейс, який визначає правила взаємодії між операційною системою та користувачем і впливає на зручність роботи. Найвідоміші операційні системи включають MS-Windows, MS-DOS, GNU/Linux, OS/2, UNIX, iOS, MacOS (рис. 1.1, рис. 1.2, рис 1.3, рис. 1.4) [17].

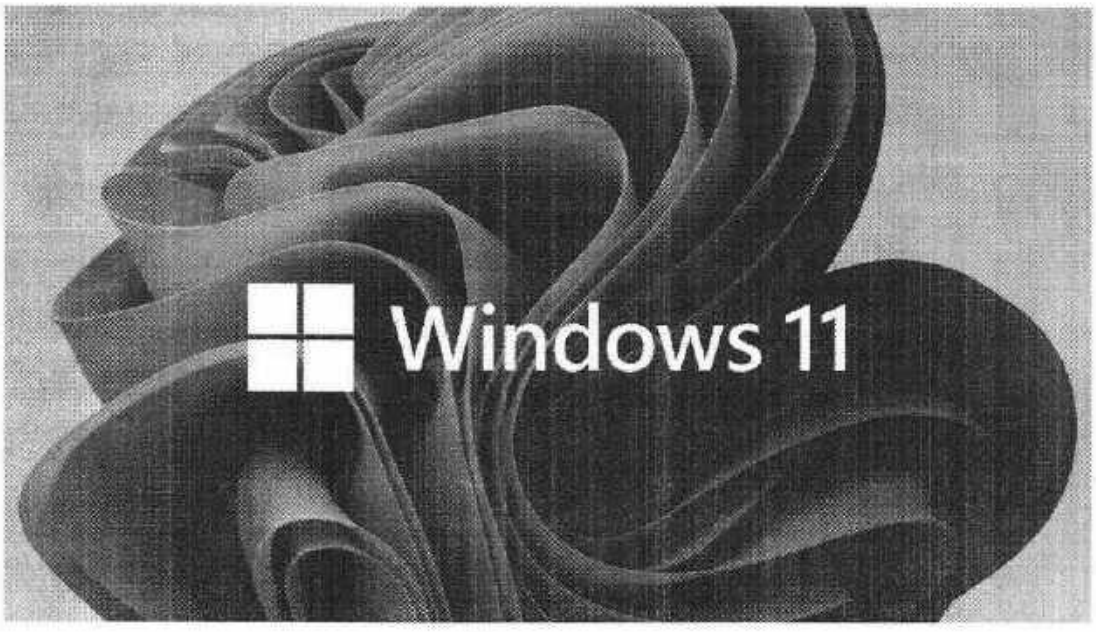


Рисунок 1.1 – Операційна система Windows [18]

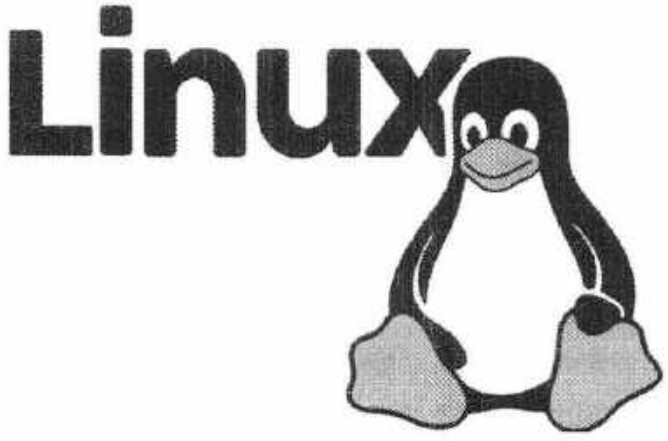


Рисунок 1.2 – Операційна система Linux [19]

Рисунок 1.3 – Операційна система MacOS [20]

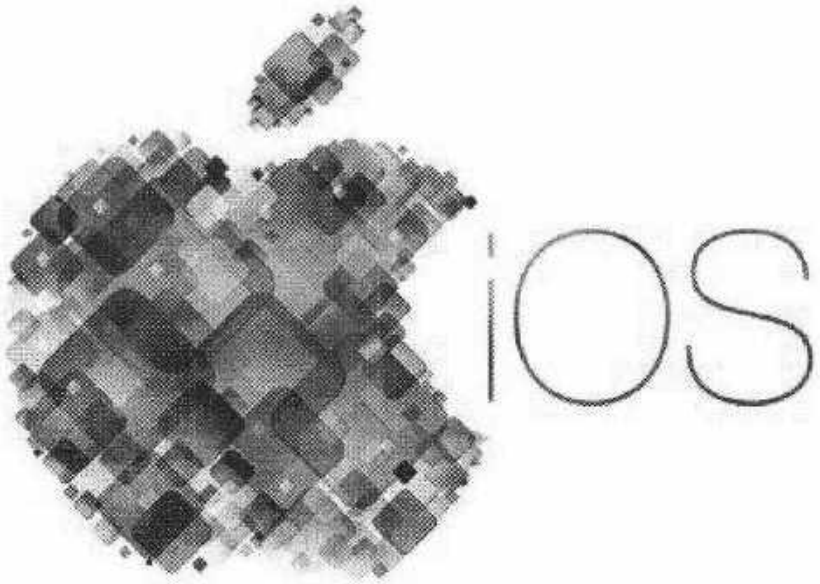


Рисунок 1.4 – Операційна система iOS [21]

Реалізація багатозадачності може розподілятися на два класи залежно від кількості одночасно виконуваних завдань в операційних системах:

- багатозадачні системи, такі як Unix, OS/2 і Windows, дозволяють виконувати декілька завдань одночасно;
- однозадачні системи, такі як MS-DOS, дозволяють виконувати лише одне завдання одночасно. Багатозадачні операційні системи повністю реалізують мультипрограмний режим, що вирішує проблеми розподілу ресурсів і конкуренції.

Багатозадачний режим, в якому використовується ідея поділу часу, називається витіскаючим (preemptive). Кожній програмі виділяється певний квант процесорного часу, після закінчення якого керування передається іншій програмі. Такі системи, як Windows, дозволяють деяким користувальницьким програмам монополізувати процесор і працювати у невитіскаючому режимі. Однак більшість комерційних операційних систем працюють у витіскаючому режимі для користувальницьких програм [22].

Операційні системи також можна класифікувати за підтримкою багатокористувацького режиму:

- однокористувацькі системи, такі як MS-DOS і Windows, дозволяють працювати тільки одному користувачеві одночасно.
- багатокористувацькі системи, такі як Windows NT і Unix, дозволяють працювати багатьом користувачам одночасно і мають механізми захисту персональних даних кожного користувача. З метою підвищення продуктивності обчислювальних систем з'явилися багатопроцесорні системи, що складаються з двох або більше процесорів, що паралельно виконують команди. Підтримка багатопроцесування є важливою властивістю операційних систем, таких як Linux, Solaris, Windows NT та інші, і призводить до ускладнення алгоритмів керування ресурсами [23].

Існують два типи багатопроцесорних операційних систем: симетричні та асиметричні. У симетричних системах кожен процесор має однакове ядро і може виконувати завдання незалежно, що призводить до повної децентралізованої обробки. У цьому випадку кожен процесор має доступ до всієї пам'яті. У асиметричних системах процесори неоднакові, зазвичай існує головний процесор (master) і підлеглі (slave), які працюють за визначенням головного процесора [24].

У багатозадачних операційних системах також існують системи реального часу, які використовуються для керування технічними об'єктами або технологічними процесами. У таких системах необхідно дотримуватися обмежень часу реакції на зовнішні події, і програми повинні обробляти дані швидше, ніж вони надходять. Архітектура систем реального часу може бути дуже жорсткою, і

вони можуть не мати віртуальної пам'яті, що дозволяє уникнути непередбачуваних затримок у виконанні програм [25].

Щодо мережевих операційних систем, вони виконують роль "мозку" мережі і забезпечують взаємодію програмного та апаратного забезпечення мережі. Мережеві операційні системи поділяються на однорангові і клієнт-серверні. У однорангових системах можна використовувати будь-який комп'ютер як робочу станцію або сервер. У таких мережах мережеві операційні системи встановлюються на кожному комп'ютері, і назва мережі є похідною від операційної системи, що утворює мережу. В однорангових мережах кожен комп'ютер може надавати свої ресурси іншим комп'ютерам у мережі. Однак, продуктивність таких мереж зменшується зі збільшенням їх розмірів і кількості взаємодій між комп'ютерами. Експлуатація та підтримка однорангових мереж є складними, оскільки адміністраторам потрібно керувати багатьма сервісами на кожній машині окремо. Користувачі таких мереж також мають можливість змінювати настройки операційних систем самостійно, що може призводити до проблем з функціонуванням всього програмного забезпечення робочої станції.

У мережах клієнт/сервер встановлюється мережна операційна система (ОС) на сервері, така як Windows 95/98, Windows 2000, Windows NT, Windows XP, Windows Millennium, Novell NetWare або UNIX. Серверний комп'ютер керує мережею та надає свої ресурси клієнтським робочим станціям. Серверна ОС відповідає за координацію використання ресурсів і послуг на сервері. Клієнтами в такій мережі можуть бути будь-які мережеві пристрої, які звертаються до сервера для отримання його ресурсів і послуг.

Для взаємодії клієнта і сервера на клієнтських комп'ютерах встановлюється відповідне клієнтське програмне забезпечення, яке підтримує загальний протокол взаємодії. У клієнт/серверних мережах користувачі реєструються зі своїх робочих станцій, повідомляючи серверу своє ім'я та пароль. Після успішної аутентифікації сервер надає користувачеві доступ до ресурсів і послуг, на які має право. Серверна ОС забезпечує надійність і безпеку збережених та оброблюваних на сервері даних.

Мережева ОС дозволяє користувачам спільно використовувати дорогі апаратні ресурси мережі, такі як принтери, сканери, дискові накопичувачі тощо.

Вона також дозволяє спільно використовувати програмне забезпечення, яке встановлено лише на сервері, а також інформаційні ресурси, наприклад, бази даних сервера. Мережева ОС дозволяє організувати сумісну роботу великого колективу користувачів з оперативним обміном інформації між ними.

Сучасні операційні системи, такі як Windows XP, Windows 2000, Windows NT Server і NetWare, складаються з керування локальними ресурсами комп'ютера, серверної частини для надання власних ресурсів та послуг загальному користуванню, клієнтської частини операційної системи для розпізнавання і переспрямування запитів до віддалених ресурсів у мережу, а також комунікаційних засобів для обміну повідомленнями в мережі.

Програма переспрямування резидентно розташовується в пам'яті комп'ютера. Вона перехоплює запити користувача або його програми до операційної системи комп'ютера, аналізує їх та спрямовує до відповідної ОС на тому ж комп'ютері або до адресованого сервера.

У мережі вузли, які представлені комп'ютерами, взаємодіють за допомогою комунікаційних протоколів, які реалізуються як програмні та апаратні засоби. Протоколи нижніх рівнів зазвичай поєднують програмні та апаратні засоби, а протоколи верхніх рівнів зазвичай базуються на програмних засобах. Крім комп'ютерів, протоколи також використовуються іншими мережевими пристроями, такими як концентратори, мости, комутатори, маршрутизатори тощо. Протокол є узгодженістю між взаємодіючими об'єктами (комп'ютерами), але це не означає, що він є стандартним [26].

1.4 Проміжне програмне забезпечення

Підприємства в сучасну епоху продовжують використовувати цифрові технології. Такі організації використовують різні апаратні та програмні продукти для досягнення своїх цілей. Апаратне та програмне забезпечення, яке використовується в організаційній структурі, було розроблено по-різному, тобто вони не обов'язково створювалися для спільної роботи. Однак така організація

потребує впевненості, що її апаратне та програмне забезпечення можуть працювати разом.

Проміжне програмне забезпечення можна визначити як програмне забезпечення, яке різні програми використовують для зв'язку один з одним. Таким чином, він діє як прихований перехідний рівень, який забезпечує керування даними та обмін даними для розподілених програм [27].

За допомогою проміжного програмного забезпечення користувачі можуть виконувати такі запити, як надсилання форм у веб-браузері. Таке програмне забезпечення також дозволяє веб-серверу повертати динамічні веб-сторінки в профілі користувача.

Проміжне програмне забезпечення дозволяє розробникам створювати додатки без створення власних інтеграцій, коли їм потрібно підключити джерела даних, компоненти додатків, пристрої або обчислювальні ресурси.

Проміжне програмне забезпечення надає різні служби та програми, які взаємодіють із платформами обміну повідомленнями, такими як передача репрезентативного стану (REST), нотація об'єктів JavaScript (JSON), розширювана мова розмітки (XML), простий протокол доступу до об'єктів (SOAP) або веб-сервіси.

Проміжне програмне забезпечення також дозволяє компонентам, написаним на різних мовах, таких як Java, Ruby, C++, PHP і Python, спілкуватися один з одним.

Розробники використовують проміжне програмне забезпечення, щоб досягти наступного:

- безпечні з'єднання та передача даних. Проміжне програмне забезпечення використовує протокол мережевої безпеки, такий як Transport Layer Security (TSL), щоб встановити захищене з'єднання між зовнішньою програмою та джерелами даних на серверній частині. Таке програмне забезпечення також може запропонувати можливості автентифікації, спонукаючи зовнішню програму запитувати цифрові сертифікати або облікові дані (електронна адреса/ім'я користувача та пароль).
- налаштовуйте та контролюйте інтеграції та підключення. Проміжне програмне забезпечення налаштовує відповідь від служби або внутрішньої

програми на основі запиту клієнта або запиту зовнішньої програми. Ідеальним прикладом може бути налаштування роздрібної торгівлі, де проміжне програмне забезпечення може використовувати інформацію про місцезнаходження в заголовку запиту HTTP або IP-адресу для сортування результатів пошуку продукту з внутрішньої бази даних інвентаризації.

- динамічно керуйте трафіком у розподілених системах. Типова система може мати коливання, наприклад кількість запитів за секунду та трафік. Корпоративне проміжне програмне забезпечення можна використовувати, коли трафік або запити зростають, щоб розподілити клієнтські запити між різними серверами. Таке рішення також стане в нагоді, коли кілька клієнтів одночасно отримують доступ до ресурсів з одного джерела даних [28].

Типова програма проміжного програмного забезпечення містить різні компоненти, які підключаються для створення конвеєра даних. Це основні компоненти архітектури проміжного ПЗ:

- консоль управління. Правила програмного забезпечення, дії системи проміжного програмного забезпечення та конфігурації знаходяться на консолі керування.
- клієнтський інтерфейс. Це зовнішня частина проміжного програмного забезпечення, яке взаємодіє з програмою. Розробники використовують функції, надані клієнтським інтерфейсом, для виконання різних дій із програмою.
- внутрішній інтерфейс проміжного ПЗ. Це як внутрішній клей, який з'єднує різні компоненти програмного забезпечення. Різні компоненти використовують цей внутрішній інтерфейс для спільної роботи та досягнення спільної мети.
- менеджер по контракту. Менеджер контрактів визначає правила обміну даними в проміжному програмному забезпеченні. Це також компонент, який гарантує, що різні програми дотримуються встановлених правил під час обміну даними.
- інтерфейс платформи. Цей інтерфейс має програмні компоненти, які працюють з різними операційними системами. Саме цей інтерфейс забезпечує сумісність проміжного програмного забезпечення з різними платформами.

- монітор виконання. Завданням монітора виконання є постійний моніторинг руху даних у системі. Цей компонент виявляє та повідомляє про незвичайну поведінку, на яку інженерам необхідно вжити заходів.
- менеджер бази даних. Менеджер баз даних може бути недоступний у всіх системах проміжного програмного забезпечення. Такий компонент інтегрується з різними типами даних.
- менеджер сесій. Такий компонент зберігає записи про дії з даними для звітності та забезпечує безперебійний потік інформації [29].

Різні типи проміжного ПЗ:

- проміжне програмне забезпечення, орієнтоване на повідомлення (MOM);
- проміжне програмне забезпечення API (інтерфейс прикладного програмування);
- проміжне програмне забезпечення віддаленого виклику процедур (RPC);
- проміжне програмне забезпечення транзакцій;
- робототехнічне проміжне програмне забезпечення;
- проміжне програмне забезпечення пристрою [30].

1.5 Висновки

Отже, у даному розділі проведено всебічний аналіз предметної області, включаючи її змістовний аналіз, структурні та функціональні особливості. Виявлено, що обчислювальна техніка та мультикомп'ютерні системи мають велике значення для різних галузей, забезпечуючи збір, обробку, зберігання та передачу інформації.

Аналіз наявного програмно-апаратного забезпечення показав, що на сьогоднішній день ЕОМ та обчислювальна техніка стали невід'ємною частиною більшості сучасних організацій, використовуються для різних цілей, включаючи збір та аналіз даних, розробку програмного забезпечення та забезпечення комунікації та зв'язку.

Аналіз системного програмного забезпечення показав значну кількість наявних архітектурних рішень, що можуть бути використані для побудови мультикомп'ютерних систем з загальною пам'яттю. Виявлено різноманітні підходи до проектування таких систем та їхньої реалізації.

Дослідження проміжного програмного забезпечення показало його важливість у забезпеченні ефективної комунікації та обміну даними між комп'ютерними вузлами мультикомп'ютерної системи. Виявлено різні підходи до розробки проміжного програмного забезпечення та його вплив на продуктивність та надійність системи.

Загальний висновок полягає у встановленні необхідності подальшого дослідження та розробки універсальної мультикомп'ютерної системи з використанням сотової топології. Дослідження показало потенціал та переваги мультикомп'ютерних систем у контексті їхнього застосування у різних галузях. Поставлення задачі на розробку такої системи ставить перед дослідниками виклики щодо вибору оптимальної архітектурної та програмної бази, а також ефективного впровадження системи з урахуванням вимог до продуктивності та надійності.

2 АРХІТЕКТУРА АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

2.1 Поняття топології та її види

При створенні комп'ютерної мережі важливо визначити топологію - спосіб розміщення мережевого обладнання та кабелів. Вибір оптимальної топології дозволить забезпечити надійну та ефективну роботу мережі, зручне керування мережевим трафіком і знизити витрати на її установку та обслуговування. Крім того, бажано, щоб мережа можна було розширювати, переходячи на швидші комунікаційні технології.

Поняття «топологія» в основному стосується положення мережевих вузлів, таких як маршрутизатори, комп'ютери, концентратори, комутатори та точки доступу. Фізична структура мережі, або топологія, визначається взаємозв'язками між цими вузлами. Вибір конкретної топології мережі визначає її характеристики, такі як:

- тип необхідного мережевого обладнання;
- функціональність цього обладнання;
- потенціал масштабованості мережі та її протоколи керування [31].

Залежно від вимог існують різні топології для побудови мережі:

- топологія сітки;
- зоряна топологія;
- деревоподібна (ієрархічна) топологія;
- топологія шини;
- кільцева топологія;
- сотова топологія.

Кільцева та сітчаста топології зручні для однорангової передачі. Зірка та дерево зручніші для клієнт-сервера. Топологія шини однаково зручна для обох з них [32-33].

Топологія сітки.

Топологія сітки - це структура або організація з'єднань між пристроями (комп'ютерами, серверами, маршрутизаторами і т. д.) у комп'ютерній мережі. У топології сітки кожен пристрій підключений безпосередньо до кожного з сусідніх пристроїв, створюючи мережу з крос-зв'язками між вузлами. Така структура нагадує мережу сітки або сітку павутини (рис 2.1).

У топології сітки відсутні централізовані вузли або вузли керування. Кожен пристрій може комунікувати безпосередньо з будь-яким іншим пристроєм у мережі. Це забезпечує високу надійність та масштабованість мережі, оскільки в разі виходу з ладу одного пристрою інші пристрої можуть продовжувати працювати незалежно.

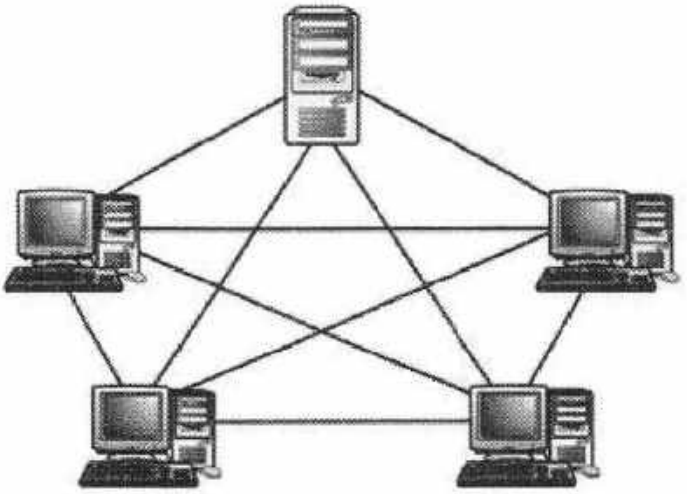


Рисунок 2.1 – Топологія сітки [34]

Топологія «зірка».

У зіркоподібній топології кабелі проходять від кожного комп'ютера до розташованого в центрі пристрою, який називається hub (хаб). Мережі з топологією «зірка» вимагають центральної точки з'єднання між сегментами медіа. Ці центральні точки називають хабами (рис 2.2).

Концентратори — це спеціальні повторювачі, які долають електромеханічні обмеження носія. Кожен комп'ютер у зірковій мережі зв'язується з центральним концентратором, який повторно надсилає повідомлення на всі комп'ютери (у широкомовній мережі) або лише комп'ютер призначення (у комутованій мережі).

Ethernet 10 base T — популярна мережа, заснована на топології «зірка».

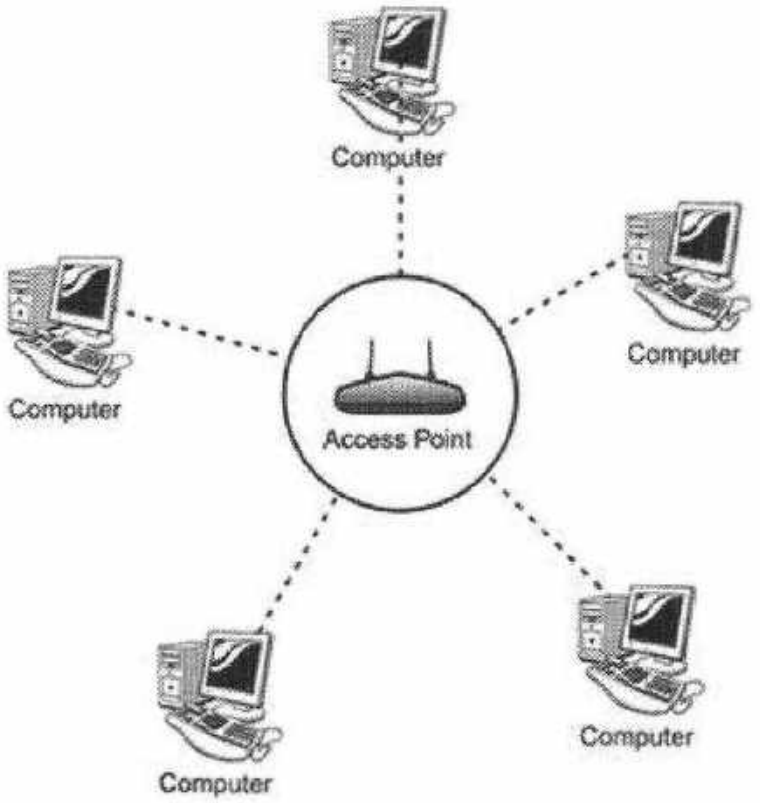


Рисунок 2.2 – Зіркоподібна топологія [35]

Деревоподібна (ієрархічна) топологія.

Це схоже на зіркову мережу, але вузли підключені до вторинного концентратора, який, у свою чергу, підключений до центрального концентратора (рис 2.3). Центральний хаб є активним хабом. Активний концентратор містить повторювач, який регенерує шаблон бітів, який він отримує, перш ніж відправляти їх. Вторинний концентратор може бути активним або пасивним. Пасивний концентратор забезпечує просте фізичне з'єднання між підключеними пристроями.

Топологія шини.

Топологія шини є однією з основних топологій комп'ютерних мереж і використовується для побудови локальних мереж (LAN). У топології шини всі

пристрої підключаються до одного спільного шинного кабелю, який служить комунікаційним каналом.

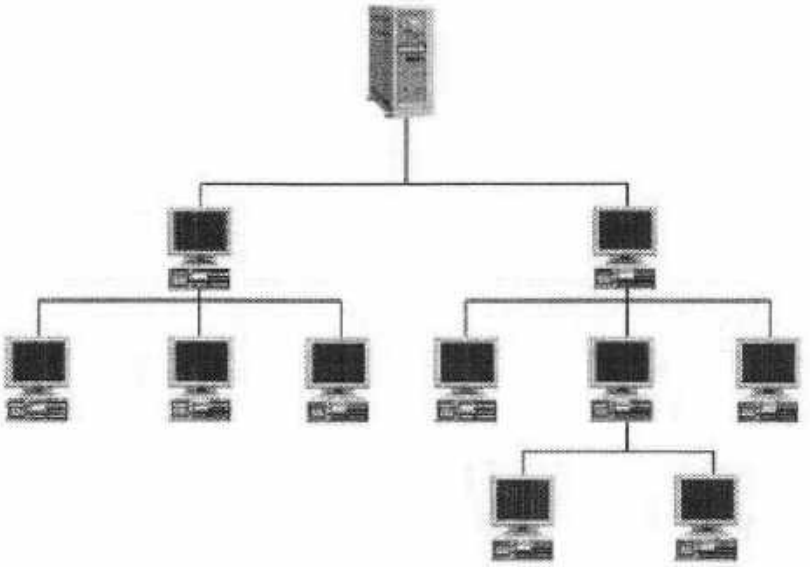


Рисунок 2.3 – Деревоподібна топологія [36]

У цій топології кожен пристрій підключений до шини, і всі дані, що передаються в мережі, пересилаються по цьому шинному кабелю (рис 2.4). Кожен пристрій слухає передавані дані і виявляє тільки ті пакети, які призначені для нього. Якщо пристрій надсилає дані, він використовує шину для передачі сигналу, який розповсюджується до всіх інших пристроїв на шині.

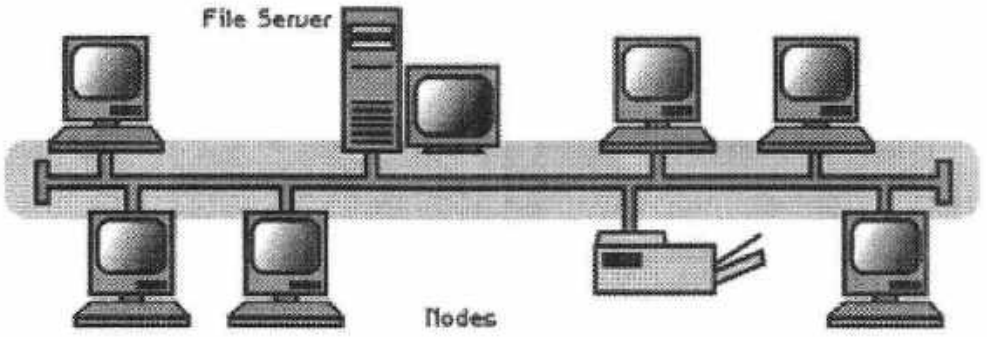


Рисунок 2.4 – Топологія шини [37]

Кільцева топологія.

У кільцевій топології кожен пристрій має спеціальну конфігурацію лінії «точка-точка» лише з двома пристроями по обидві сторони від неї (рис 2.5).

Сигнал передається по кільцю в одному напрямку, від пристрою до пристрою, поки не досягне місця призначення.

Кожен пристрій у кільці має повторювач. Коли пристрої отримують сигнал, призначений для іншого вузла, він просто регенерує біти та передає їх.

Кільцева мережа передає маркер.

Токен - це коротке повідомлення з електронною адресою одержувача.

Кожній платі мережевого інтерфейсу надається унікальна електронна адреса, яка використовується для ідентифікації комп'ютера в мережі.

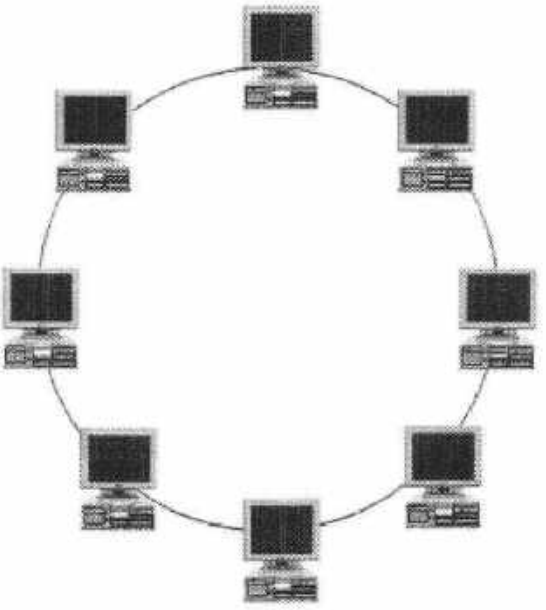


Рисунок 2.5 – Кільцева топологія [38]

Сотова топологія.

Сотова топологія, також відома як топологія комірок, є основною топологією для мобільних телефонних мереж. Вона базується на концепції розбиття місцевості на географічні області, які називаються комірками, і кожна комірка має свою базову станцію для забезпечення зв'язку з мобільними пристроями (рис 2.6).

У сотовій топології кожна комірка має свою власну базову станцію, яка керує передачею сигналів між мобільними пристроями в комірці та мережею оператора.

Базова станція приймає сигнали від мобільних пристроїв і пересилає їх до мережі, а також передає сигнали від мережі до мобільних пристроїв у комірці.

Кожна комірка має свою унікальну частоту, що дозволяє багатьом пристроям одночасно працювати у різних комірках без перешкод. Коли мобільний пристрій пересувається з однієї комірки в іншу, зв'язок передається з базової станції однієї комірки до базової станції іншої комірки, щоб забезпечити безперебійну передачу сигналу.

Особливості сотової топології:

- масштабованість: мобільні мережі можуть бути розширені шляхом додавання нових комірок, що дозволяє підтримувати зростаючу кількість користувачів;
- мобільність: завдяки сотовій топології користувачі можуть здійснювати рух і пересуватися між комірками, не втрачаючи зв'язку;
- резервування ресурсів: частотні ресурси можуть бути ефективно розподілені між різними комірками, що дозволяє більше пристроїв працювати одночасно без перешкод;
- керування рухом: мережа здатна керувати рухом мобільних пристроїв і забезпечувати передачу зв'язку при переході між комірками.

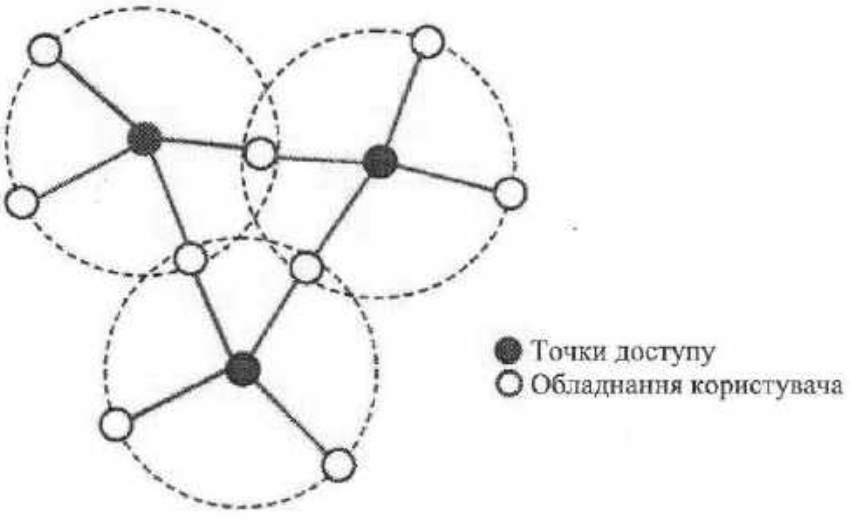


Рисунок 2.6 – Сотова топологія [39]

2.2 Вибір сотової топології

У даній роботі було обрано сотову топологію для подальшого дослідження. Давайте розглянемо її детальніше.

Сотова мережа або мобільна мережа є бездротовою мережею, яка забезпечує зв'язок між вузлами. Вона складається з осередків, що обслуговуються фіксованими приймально-передавальними пристроями (базовими станціями), розташованими на певних місцях. Ці базові станції забезпечують покриття зони, де можна передавати голос, дані та інші типи вмісту. Для забезпечення якісного обслуговування в кожній клітинці і уникнення перешкод, осередки використовують різні набори частот.

Загалом, осередки забезпечують радіопокриття широкої географічної території, що дозволяє мобільним пристроям спілкуватися між собою та зі стаціонарними трансиверами та телефонами у будь-якому місці мережі через базові станції. Навіть якщо деякі з трансиверів переміщуються через кілька комірок під час передачі, зв'язок залишається збереженим.

Сотові мережі мають декілька привабливих характеристик:

- вони мають більшу ємність порівняно з одним великим передавачем, оскільки одна і та ж частота може використовуватися для кількох каналів, якщо вони знаходяться в різних стільниках;
- мобільні пристрої споживають менше енергії, ніж один передавач або супутник, оскільки стільникові вежі розташовані ближче;
- сотові мережі забезпечують більшу зону покриття, ніж один наземний передавач, оскільки можна додавати нескінченну кількість додаткових стільникових веж, не обмежених горизонтом;
- можна використовувати сигнали вищої частоти, що дозволяє мати більшу доступну смугу пропускання та вищу швидкість передачі даних, ніж ті, які не можуть поширюватися на великі відстані;
- завдяки стисненню та мультиплексуванню даних, високочастотний сигнал може переносити кілька відео- та аудіоканалів, включаючи цифрове відео, на одній широкосмуговій хвилі [40-41].

Система сотового радіозв'язку використовує розділення земельної ділянки, яка повинна бути покрита радіосигналом, на клітини, що відповідають рельєфу місцевості та характеристикам прийому. Візерунки цих клітин мають приблизну форму регулярних фігур, таких як шестикутники, квадрати або кола, але шестикутні клітини є найбільш поширеними. Кожній клітині призначається декілька частот (f1 - f6), які використовують відповідні базові радіостанції. Групу частот можна повторно використовувати в інших клітинах, за умови, що ці самі частоти не використовуються в сусідніх клітинах, щоб уникнути перешкод у одному каналі.

Висока пропускна здатність стільникових мереж у порівнянні з мережею з одним клієнтом досягається завдяки системі комутації стільникового зв'язку, розробленій Амосом Джоелем з Bell Labs. Ця система дозволяє кільком абонентам в одному районі використовувати одну частоту, перемикаючи дзвінки на найближчу доступну вежу стільникового зв'язку на вільній частоті. Ця стратегія ефективна, оскільки одну радіочастоту можна повторно використовувати в іншій зоні для незалежного мовлення. З іншого боку, один передавач може обробити лише одну передачу на певній частоті. Проте існує певний рівень перешкод сигналу від інших стільників, які використовують ту саму частоту. Таким чином, у системі множинного доступу зі стандартним частотним поділом (FDMA) має бути проміжок між осередками, які повторно використовують ту саму частоту [42].

Візьмемо, наприклад, компанію таксі, де кожна радіостанція має ручний вибір каналу для налаштування на різні частоти. Водії перемикають канали під час руху. Вони знають, що певний діапазон частот охоплює певну територію. Якщо вони не отримають сигнал від передавача, вони спробують інші канали, доки не знайдуть той, який працює. Таксисти спілкуються між собою лише за вказівкою оператора базової станції. Це форма множинного доступу з тимчасовим поділом (TDMA) (рис 2.7).

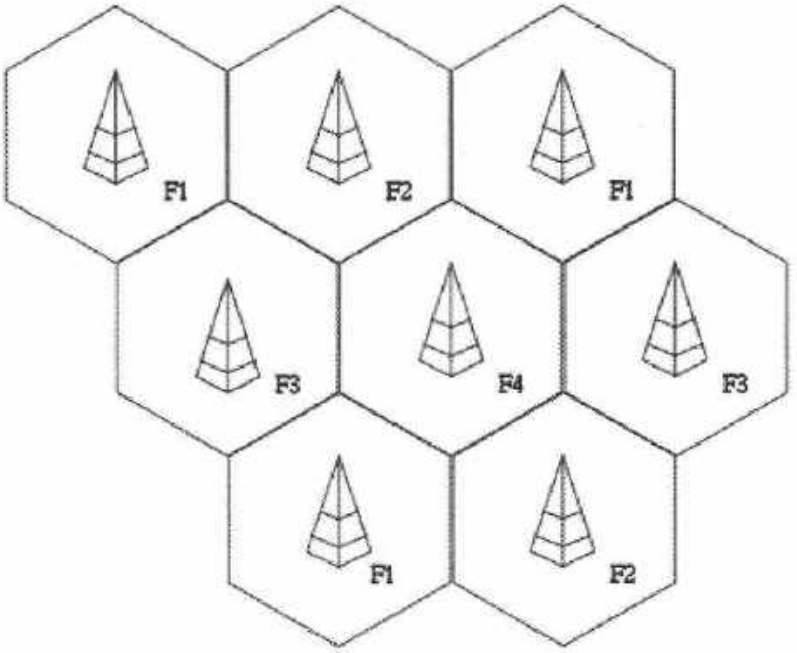


Рисунок 2.7 - Приклад коефіцієнта або шаблону повторного використання частоти [43]

2.3 Вибір архітектури мультикомп'ютерної системи

Для реалізації мультикомп'ютерної системи згідно сотової топології знадобиться наступне апаратне забезпечення:

- 3 комп'ютери (ноутбуки);
- 3 точки доступу;
- 3 антени;
- кабель вита пара.

2.3.1 Ноутбуки для мультикомп'ютерної системи

Для коректної роботи системи за відповідними параметрами було обрано 3 однакових ноутбуків, які мають наступні характеристики (рис 2.8):

- процесор Intel Core i5-1235U (3.3 - 4.4 ГГц);
- 16 Гб оперативної пам'яті;
- твердотілий накопичувач на 512 Гб;

- підтримка Wi-Fi та Bluetooth.

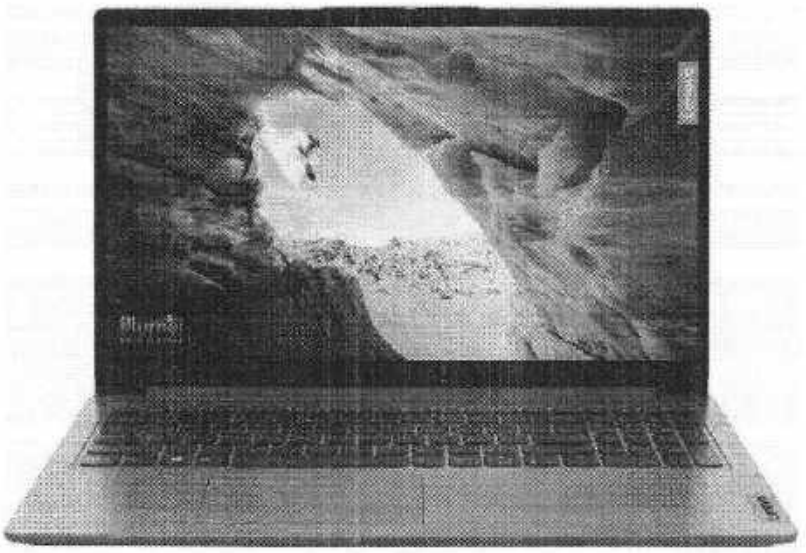


Рисунок 2.8 – Ноутбук для мультикомп'ютерної системи [44]

2.3.2 Точки доступу та антени

Загалом будь-який комп'ютер або пристрій у мережі, до якого користувачі можуть отримати доступ, можна назвати точкою доступу (Access Point, далі - AP), приклад якої зображено на рисунку 2.9.

У більшості випадків точка доступу є базовою станцією в бездротовій локальній мережі. Хоча існують інші бездротові технології, які використовують точки доступу, цей термін зазвичай відноситься до мережі Wi-Fi. Точки доступу (AP) можуть бути автономними пристроями, які підключаються до маршрутизатора або комутатора; однак функції точки доступу також вбудовані в бездротовий маршрутизатор, який широко використовується в більшості будинків і невеликих офісів.

В організації можна розгорнути кілька точок доступу, і користувачі, які перебувають у роумінгу зі своїми мобільними пристроями, переключаються з однієї точки доступу (AP) на іншу.

Назва точки доступу = ідентифікатор набору послуг (SSID)

Назви мереж Wi-Fi призначає користувач або адміністратор мережі. Коли пристрої шукають мережі Wi-Fi, вони відображають назви, які називаються «ідентифікаторами набору послуг» (SSID), усіх точок доступу Wi-Fi поблизу.

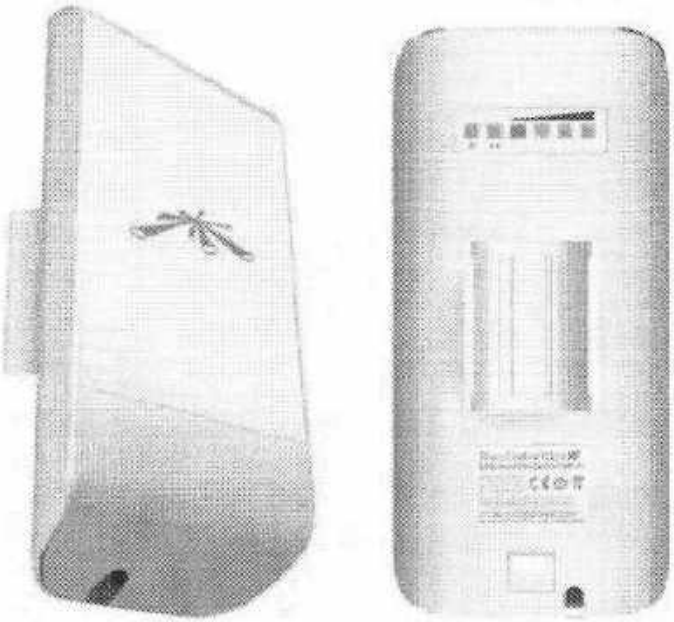


Рисунок 2.9 – Приклад точки доступу [45]

Для реалізації потрібної конфігурації мультикомп'ютерної системи було обрано 3 однакових точки доступу Ubiquiti Rocket 5ac (R5AC-Lite), що мають наступні параметри (рис 2.10):

- частота роботи Wi-Fi – 5 ГГц;
- швидкість Wi-Fi – 450 Мбіт/сек;
- живлення – PoE (Power over Ethernet).

Ubiquiti Rocket 5ac (R5AC-Lite) є бездротовим мережевим пристроєм, розробленим компанією Ubiquiti Networks. Він входить до серії Rocket, яка спеціально призначена для побудови високошвидкісних бездротових мереж з використанням технології 802.11ac.

Rocket 5ac (R5AC-Lite) пропонує високу продуктивність і широкий діапазон функцій, що робить його ідеальним рішенням для побудови точок доступу в

мережах з великою кількістю користувачів або віддалених мереж з високою пропускною здатністю.

Рисунок 2.10 – Точка доступу Ubiquiti Rocket 5ac (R5AC-Lite) [46]

Але, дані точки доступу не мають антен, тому було прийнято рішення обладнати їх антенами, які мають кругову діаграму направленості.

Було обрано Ubiquiti AirMax Omni 5G (AMO-5G10) - це всенаправлена антена, спроектована для використання з точкою доступу Rocket M5 в 5 ГГц частотному діапазоні (рис 2.11). При спільному використанні даного обладнання можна покрити сигналом територію в радіусі 360 градусів на відстані не більше 2 км, що забезпечує гнучкість налаштування і зручність у використанні. Дана модель антени забезпечує посилення сигналу 10 dbi.

Ubiquiti AirMax Omni 5G (AMO-5G10) пропонує високу ефективність, що робить її ідеальним вибором для розгалужених бездротових мереж, точок доступу, а також для забезпечення зв'язку на великих відкритих просторах.

Рисунок 2.11 – Антена Ubiquiti AirMax Omni 5G (AMO-5G10) [47]

2.3.3 Налаштування Rocket як точки доступу

Розпаковуємо пристрій та дотримуємося інструкції для його підключення. Перед підключенням блоку живлення до мережі, ми перевіряємо правильність підключення порту кабелю PoE, оскільки неправильне підключення може завдати шкоди мережевому обладнанню та ПК.

Щоб налаштувати ПК для мережевого з'єднання, встановлюємо IP-адресу 192.168.1.222 (можливий будь-який в діапазоні 1-19 та 21-254) та маску мережі 255.255.255.0. Немає потреби вказувати шлюз та DNS (рис 2.12).

З'єднуємо LAN-порт блоку живлення Rocket-AP з уже налаштованою мережевою карткою ПК за допомогою звичайного патч-корду.

Для доступу до налаштувань, в рядку адреси будь-якого з сучасних браузерів ПК вводимо адресу 192.168.1.20. Якщо підключення виконано правильно, то з'являється зображення, як на рисунку 2.13.

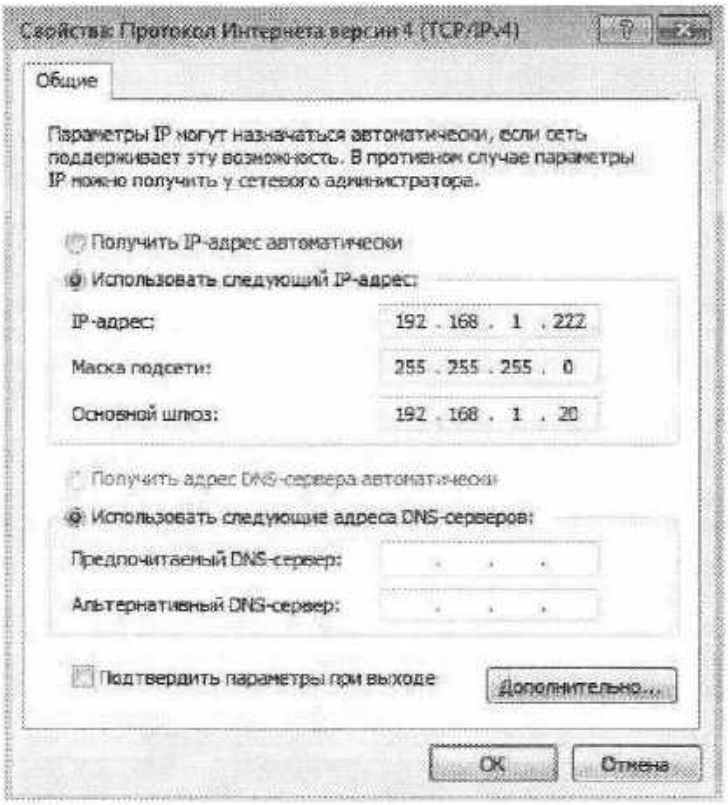


Рисунок 2.12 – Налаштування IP-адреси

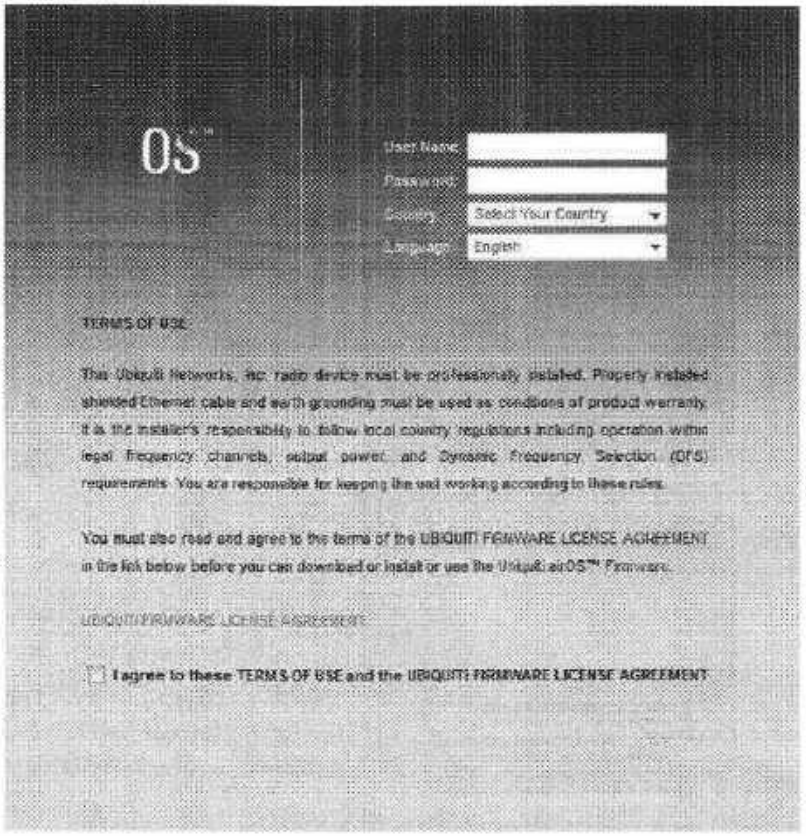


Рисунок 2.13 – Вікно авторизації для входу в налаштування [48]

Для входу в систему вводимо слово "ubnt" як логін і пароль. У третьому полі вибираємо країну використання. Цей вибір визначає максимально допустиму потужність передавача, оскільки у кожній країні існують різні обмеження (рис 2.14).

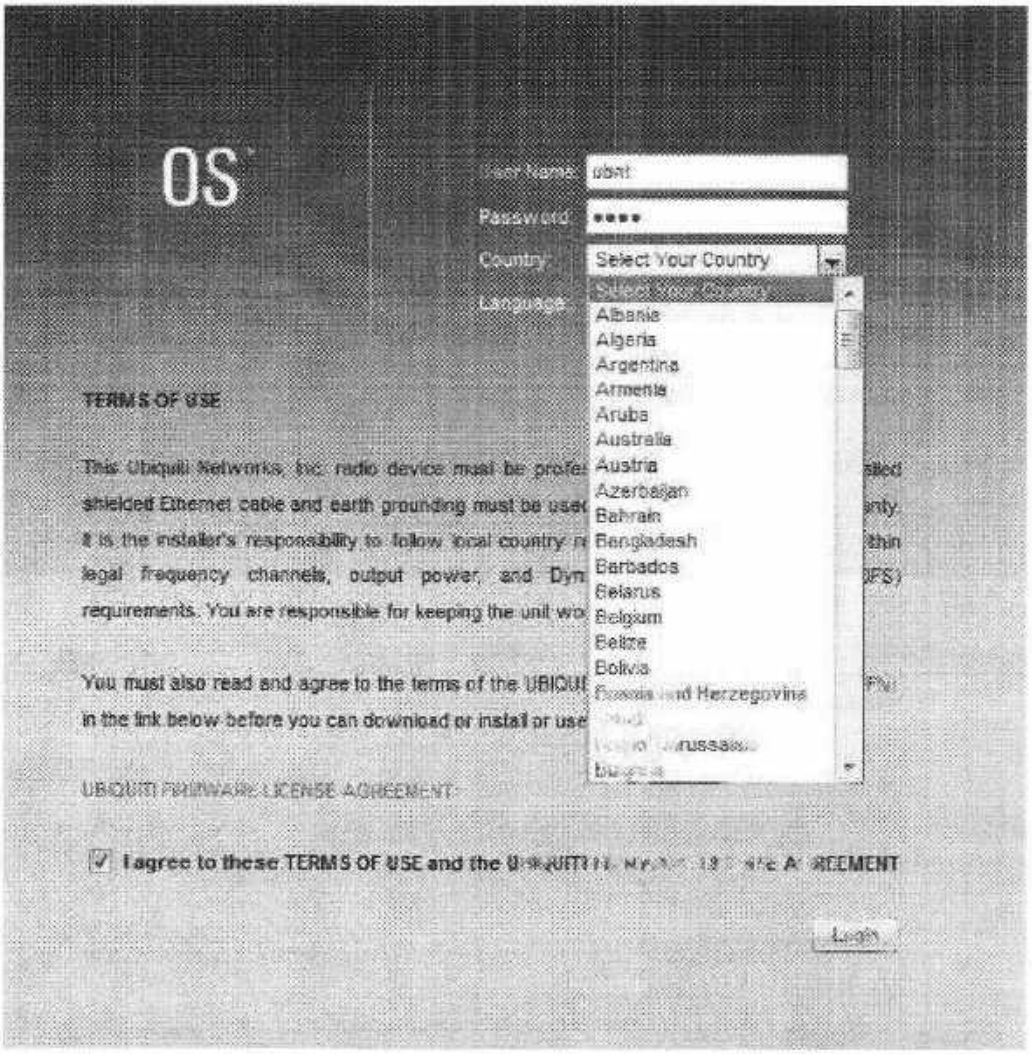


Рисунок 2.14 – Вхід у налаштування

Перш ніж розпочати налаштування, необхідно провести аналіз радіочастотного ефіру з використанням утиліти AirView. Цей аналіз допоможе вибрати найбільш вільний від сторонніх сигналів радіоканал. У правому верхньому куті відкриваємо список і вибираємо AirView.

Після цього з'являється вікно з попередженням, що для роботи AirView необхідна встановлена Java (рис 2.15).

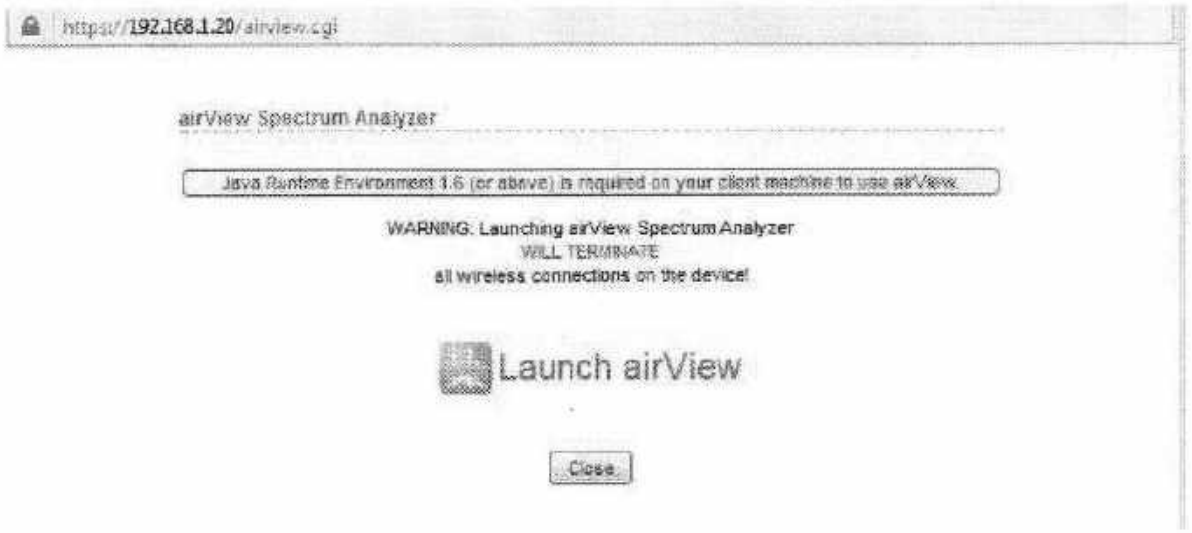


Рисунок 2.15 – Попередження про наявність Java

Після цього натискаємо кнопку "Launch AirView", що запустить аналізатор спектру AirView.

На основі результатів, отриманих з AirView, можемо зробити висновок, що район 1 та 9 каналів є менш забрудненим, а джерела найбільшої інтенсивності розташовані на 10-12 каналах. Обираємо 1 канал і запам'ятовуємо вільний діапазон частот 2410-2414 МГц (рис 2.16).

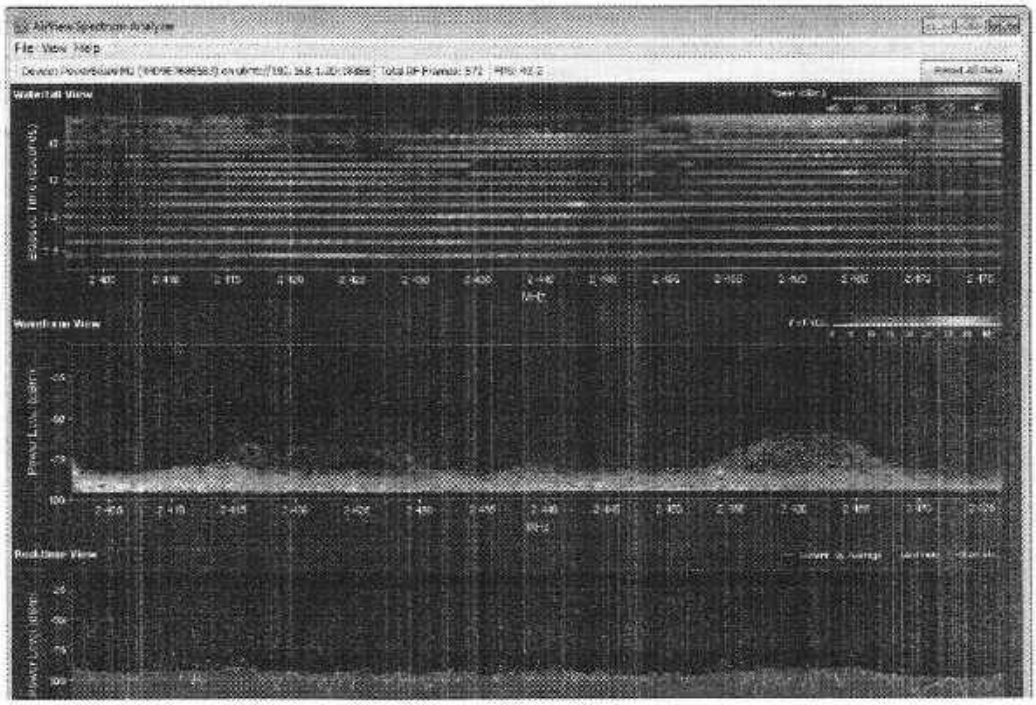


Рисунок 2.16 – Аналізатор спектру AirView [49]

Повертаємось до вкладки Network в AirOS інтерфейсу та переходимо в режим налаштувань, який може бути простим (за замовчуванням) або просунутим. Більшість користувачів можуть скористатись простим режимом (рис 2.17).

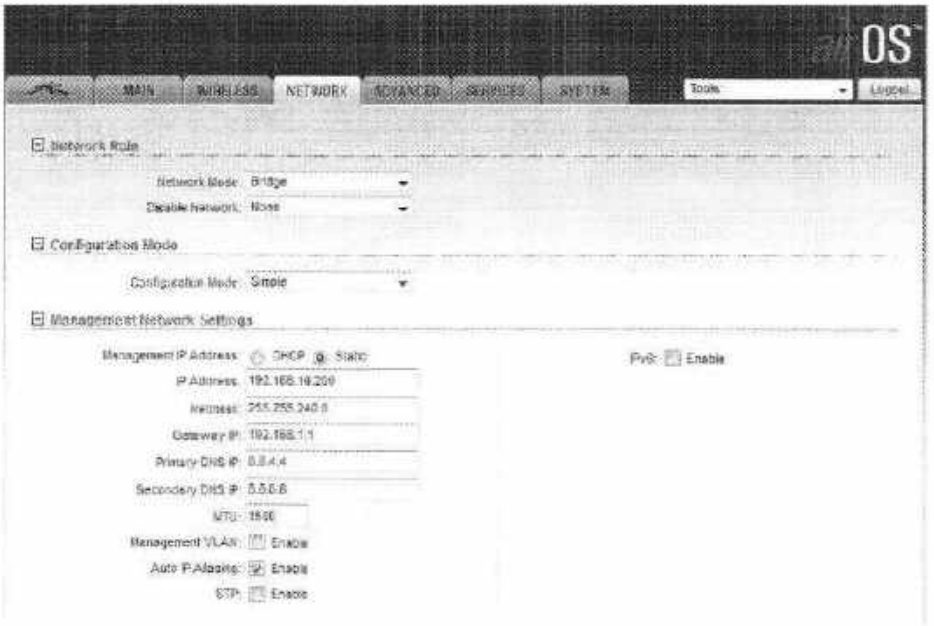


Рисунок 2.17 – Налаштування точки доступу [50]

В розділі Network mode вибираємо режим моста - Bridge. Необхідно пам'ятати, що другий параметр Disable Network не повинен бути вимкненим, оскільки це може призвести до відключення LAN/WAN портів. У третьому розділі, який має назву Management Network Settings, потрібно вказати налаштування мережі для Rocket-AP.

Є два можливих варіанти: а) використання режиму DHCP, коли Rocket-AP отримує IP-адресу від DHCP-сервера. В цьому випадку необхідно вказати резервну IP-адресу та маску, щоб забезпечити доступність Rocket-AP, якщо він не отримає IP-адресу від DHCP-сервера. Решта пристроїв отримають свої IP-адреси від DHCP-сервера; б) використання статичної IP-адреси (режим Static), коли Rocket-AP завжди доступний за однією адресою, і робота DHCP-серверів його не стосується. У цьому випадку необхідно вказати IP-адресу, маску, шлюз та DNS-сервери. Якщо пристрій використовується для доступу в Інтернет, то найкращим варіантом буде вказати DNS-сервери Вашого провайдера як Primary DNS, а Secondary DNS -

публічний, наприклад, від Google (8.8.8.8). Інші параметри можна залишити без змін.

Рекомендується змінювати IP адресу пристрою зі значення за замовчуванням та розташовувати його в іншій підмережі. Наприклад, для Rocket-AP можна використати IP адресу 192.168.10.200. Якщо потрібно, щоб Rocket-AP був підключений до інтернету, необхідно вказати реальний шлюз як gateway, а якщо він знаходиться в іншій підмережі, то потрібно розрахувати маску. Наприклад, якщо шлюз знаходиться на 192.168.1.1, то можна використовувати маску 255.255.240.0. Після внесення змін необхідно натиснути кнопку Change, а потім Apply зправа вгорі. Пристрій перезавантажиться і браузер автоматично перенаправить на нову адресу.

Щоб відкрити сторінку на новій адресі, необхідно змінити мережеві налаштування ПК на ту саму мережу: наприклад, в нашому випадку можна використати IP 192.168.10.11 та маску 255.255.240.0. Після авторизації на пристрої потрібно відкрити вкладку Wireless (рис 2.18).

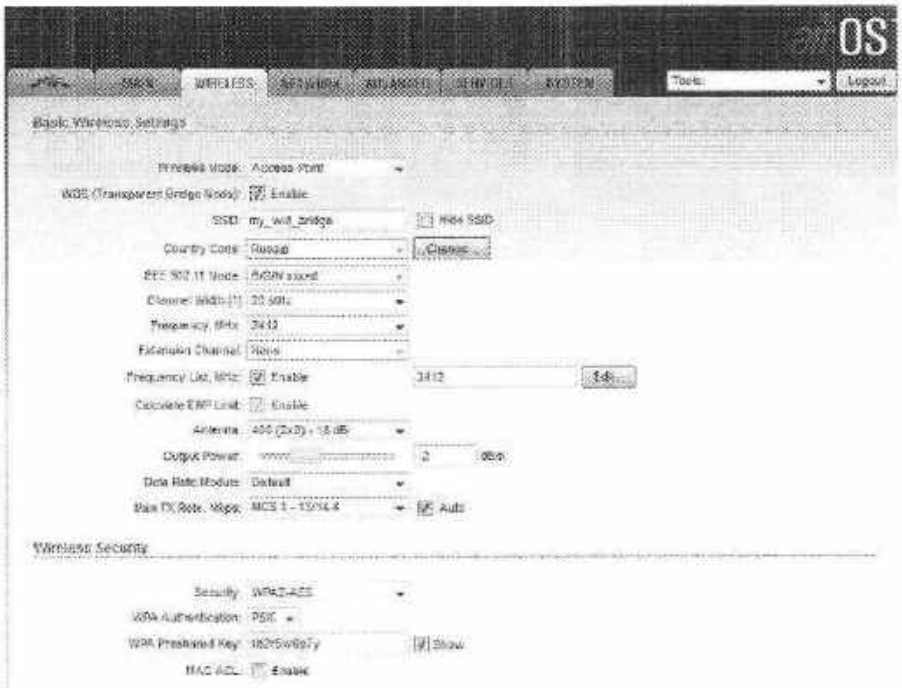


Рисунок 2.18 – Розділ Basic Wireless Settings [51]

В розділі Basic Wireless Settings задаються головні налаштування бездротової мережі.

Виставляємо налаштування згідно рисунку 2.19:

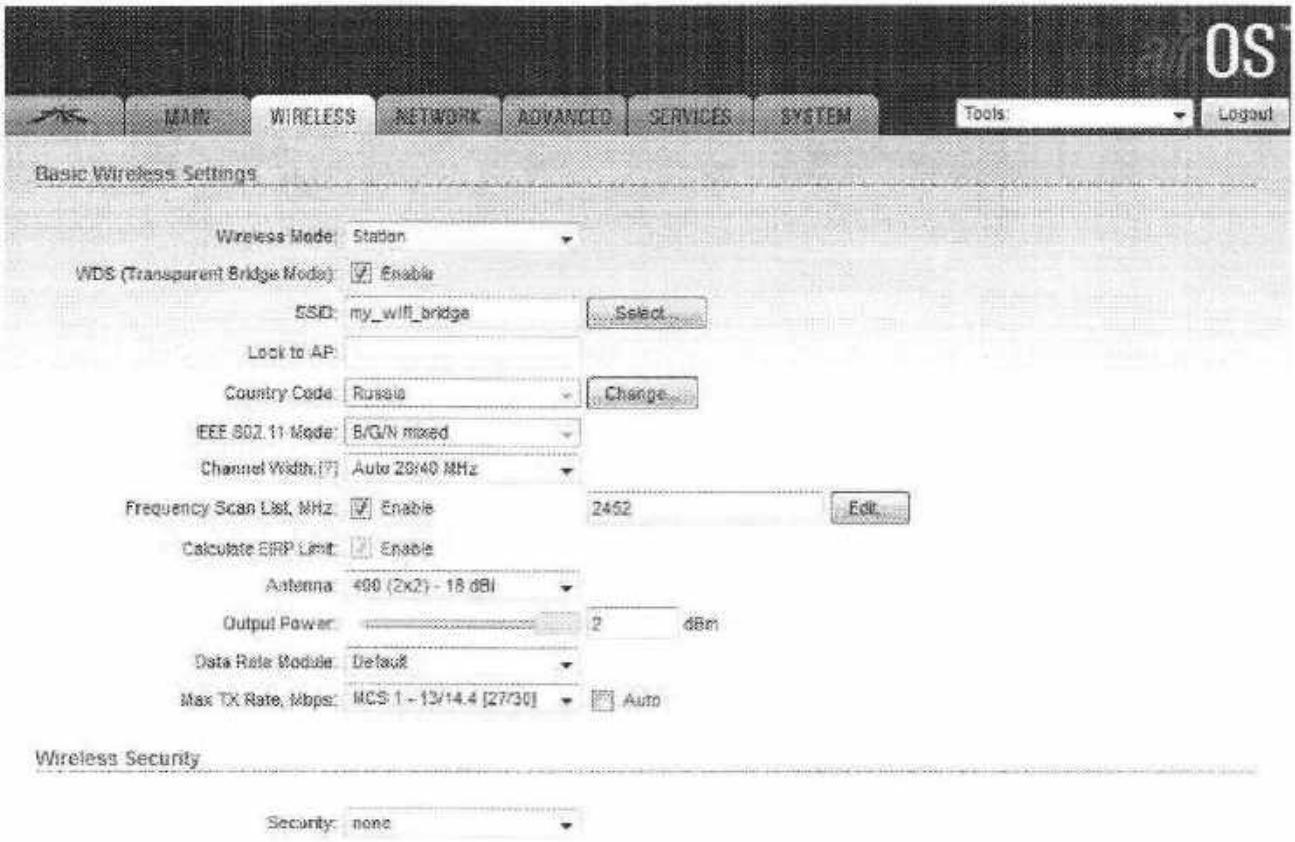


Рисунок 2.19 – Виставлені параметри у розділі Basic Wireless Settings [52]

Далі переходимо до розділу Advanced (рис 2.20). Тут можна залишити більшість налаштувань за замовчуванням і не змінювати їх, крім параметру lan speed. Більшість сучасних Ethernet-мереж працюють у дуплексному режимі на швидкості 100 Мбіт/с, тому можна в цьому списку вибрати 100 Мбіт/с Full, хоча залишення його в автоматичному режимі не є помилкою. Параметр Distance також можна залишити в автоматичному режимі.

Розділ Services залишаємо без змін. Вкладка "System" містить кілька налаштувань, які можна змінити. Рекомендовано змінити часовий пояс в полі "Time Zone" і, з міркувань безпеки, змінити ім'я користувача адміністратора на "Administrator User Name". Також, якщо потрібно, можна змінити назву пристрою, наприклад, на "Rocket Access Point" (рис 2.21).

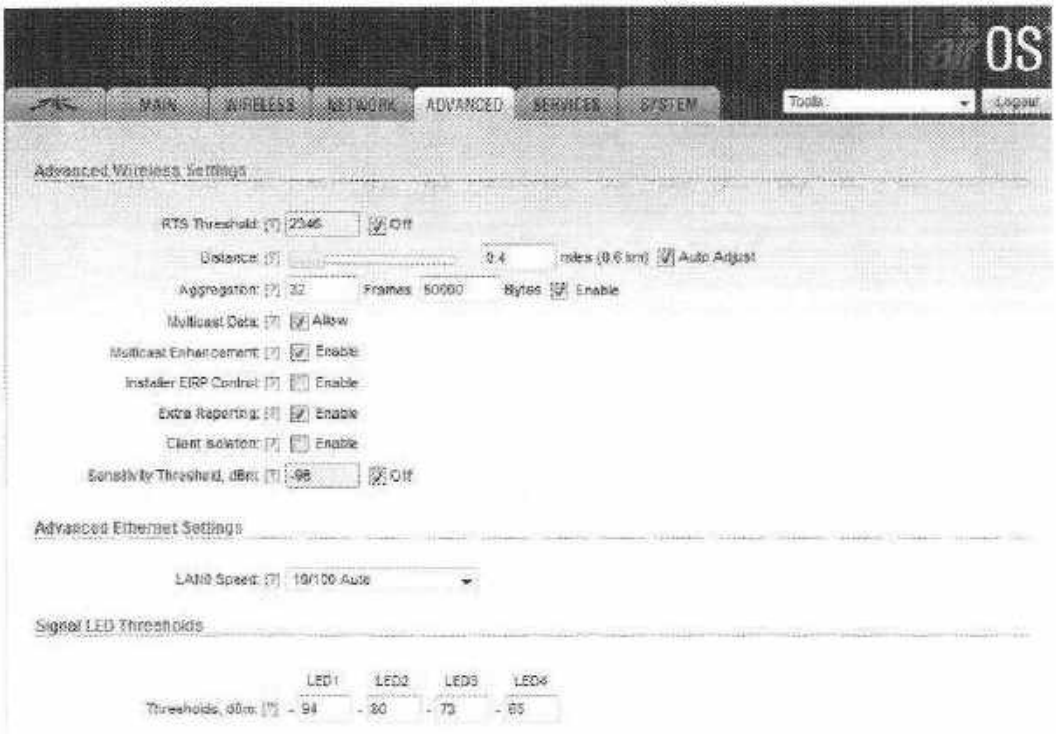


Рисунок 2.20 – Розділ Advanced [53]

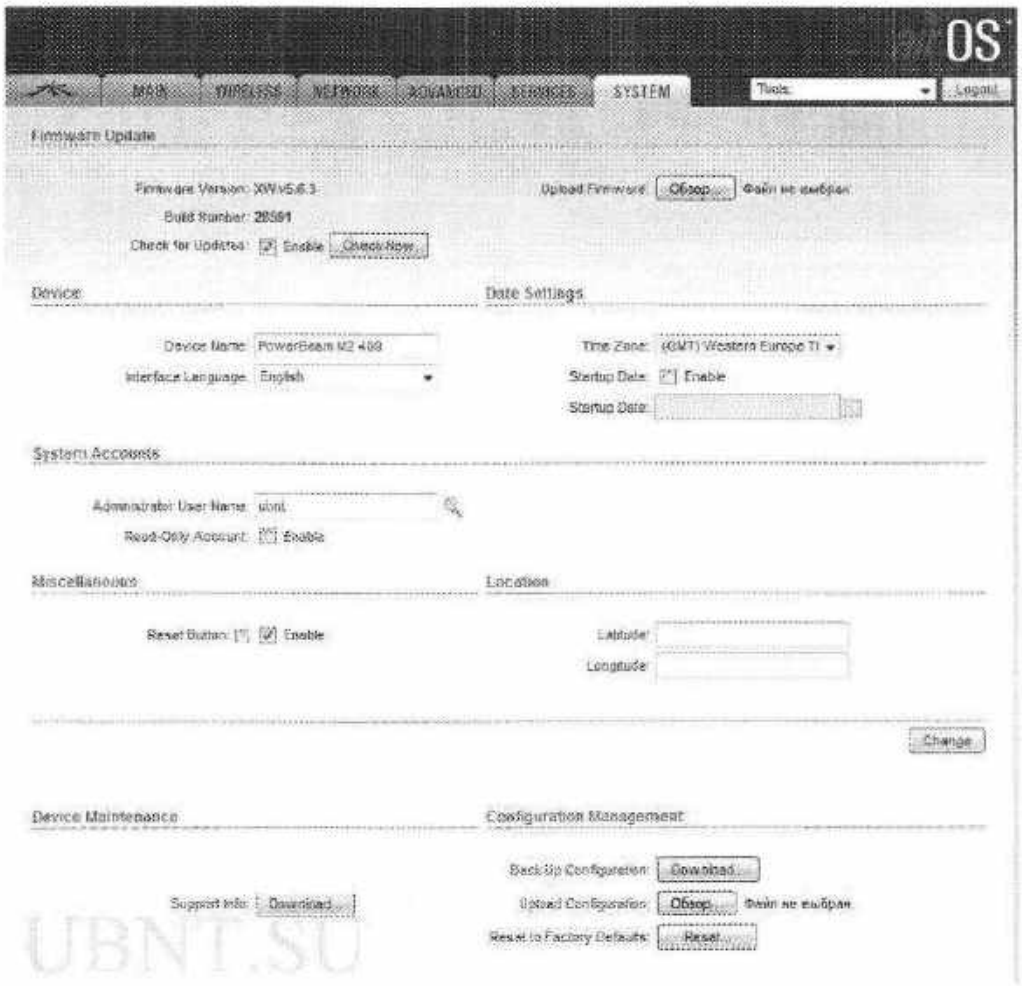


Рисунок 2.21 – Розділ System [54]

На вкладці AirMax слід включити функцію пріоритезації AirMax та вибрати рівень High зі списку. Застосування технології AirMax може помітно покращити стабільність та швидкість роботи мережі, проте ця технологія є ексклюзивним розробкою Ubiquiti, тому інші виробники пристроїв не зможуть працювати в такій мережі. Не слід змінювати налаштування Airview Port (рис 2.22).

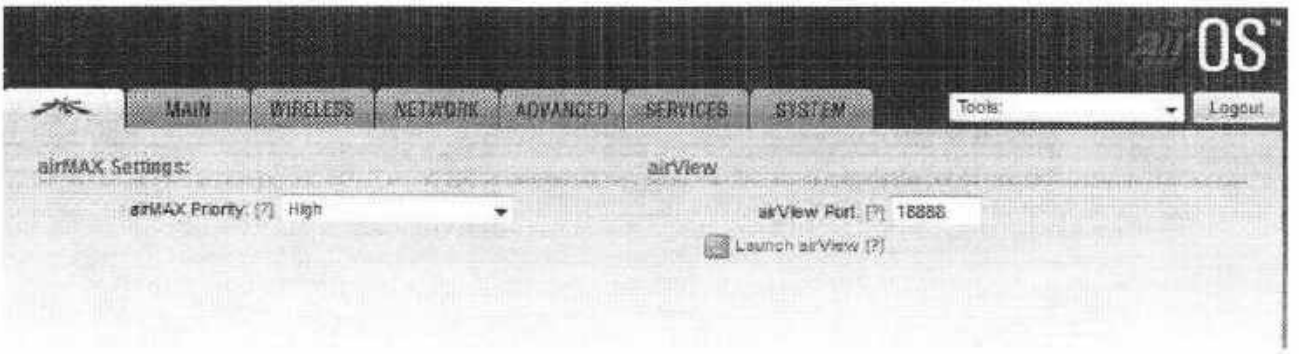


Рисунок 2.22 – Розділ AirMax [55]

Зберігаємо налаштування, перезавантажуємо пристрій.

2.4 Висновки

Отже, у цьому розділі було розглянуто поняття «топології» та яких видів вона існує, описано всі особливості, переваги та недоліки саме сотової топології, що у подальшому допомогло з вибором апаратного забезпечення, підібрано характеристики обладнання, які задовольняють стабільній роботі мультикомп'ютерній системі.

3 АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

3.1 Вибір програмних засобів

Проміжне програмне забезпечення може бути написане різними мовами програмування, однією з яких є C++ — об'єктно-орієнтована мова програмування загального призначення. Його (C++) створив Б'ярн Страуструп у Bell Labs приблизно в 1980 році. C++ дуже схожий на C (винайдений Деннісом Річі на початку 1970-х років). C++ настільки сумісний із C, що він, ймовірно, скомпілює понад 99% програм на C, не змінюючи жодного рядка вихідного коду. Хоча C++ є значною мірою добре структурованою та безпечнішою мовою, ніж C, оскільки вона базується на ООП.

Деякі комп'ютерні мови написані з певною метою. Наприклад, спочатку Java була розроблена для керування тостерами та іншою електронікою. C був розроблений для програмування ОС. Паскаль був розроблений для навчання правильним технікам програмування. Але C++ є мовою загального призначення. Він цілком заслуговує на загальновизнане прізвисько «швейцарський кишеньковий ніж мов».

Чи є C++ найкращою мовою програмування? Відповідь залежить від точки зору та вимог. Деякі завдання можна виконати на C++, але не дуже швидко. Наприклад, проектування екранів GUI для додатків.

Інші мови, такі як Visual Basic, Python, мають вбудовані елементи дизайну GUI. Тому вони краще підходять для завдань типу GUI.

Деякі з мов сценаріїв, які надають додаткам додаткові можливості програмування. Такі як MS Word і навіть Photoshop, як правило, є варіантами Basic, а не C++.

C++ все ще широко використовується, і найвідоміше програмне забезпечення базується саме на цій мові.

Хто використовує C++? Деякі з найпоширеніших сучасних систем мають критичні частини, написані на C++.

Приклади:

- Amadeus (продаж авіаквитків);
- Bloomberg (фінансове формування);
- Amazon (веб-торгівля), Google (веб-пошук);
- Facebook (соціальні мережі).

Багато мов програмування залежать від продуктивності та надійності C++ у своїй реалізації. Приклади:

- віртуальні машини Java;
- інтерпретатори JavaScript (наприклад, Google V8);
- браузери (наприклад, Internet Explorer, Firefox від Mozilla, Safari від Apple і Chrome від Google);
- додатки та веб-платформи (наприклад, платформа веб-служб Microsoft .NET).

Програми, які включають локальні та глобальні мережі, взаємодію з користувачем, числові, графічні та доступ до бази даних значною мірою залежать від мови C++.

Ось п'ять основних концепцій C++:

- змінні, що є основою будь-якої мови програмування;

Змінна — це лише спосіб зберегти деяку інформацію для подальшого використання. Ми можемо отримати це значення або дані, звернувшись до «слова», яке описуватиме цю інформацію.

Після того, як вони оголошені та визначені, змінні можуть використовуватися багато разів у тій області, в якій вони були оголошені.

- керуючі структури C++;

Коли програма виконується, код зчитується компілятором рядок за рядком (зверху вниз і здебільшого зліва направо). Це відоме як «потік коду».

Коли код зчитується зверху вниз, він може зіткнутися з точкою, де потрібно прийняти рішення. Залежно від рішення програма може перейти до іншої частини коду. Це може навіть змусити компілятор повторно запустити певний фрагмент або просто пропустити купу коду.

Таким же чином комп'ютерна програма має набір строгих правил, які визначають потік виконання програми.

- структури даних C++;

Структура даних — чудовий спосіб обійти необхідність створення тисяч змінних. C++ містить багато типів вбудованих структур даних. Найчастіше використовуються масиви.

- синтаксис C++;

Синтаксис — це компонування слів, виразів і символів.

Наприклад, адреса електронної пошти має чітко визначений синтаксис. Знадобиться деяка комбінація літер, цифр, потенційно з підкресленням (_) або крапками (.) між ними, за якими йде символ ставки (@), а потім певний домен веб-сайту (gmail.com).

Отже, синтаксис мови програмування майже однаковий. Він являють собою певний чітко визначений набір правил, які дозволяють створити деяку частину програмного забезпечення, що добре працює.

Але, якщо не дотримуватись правил мови програмування або синтаксису, отримаємо помилки.

- інструменти C++.

У реальному світі інструмент — це щось (зазвичай фізичний об'єкт), що допомагає вам швидко виконати певну роботу.

Це також стосується світу програмування. Інструмент у програмуванні — це частина програмного забезпечення, яка при використанні з кодом дозволяє програмувати швидше.

Ймовірно, існують десятки тисяч, якщо не мільйони різних інструментів на всіх мовах програмування.

Найбільш важливим інструментом, на думку багатьох, є IDE, інтегроване середовище розробки. IDE — це програмне забезпечення, яке значно спростить ваше кодування. IDE гарантують, що ваші файли та папки впорядковані, і надають гарний і зрозумілий спосіб їх перегляду.

Нижче наведено приклади використання мови С++ в різних сферах:

- операційні системи:

незалежно від того, чи це Microsoft Windows, чи Mac OSX, чи Linux – усі операційні системи мають деякі частини, запрограмовані на С++. Це основа всіх відомих ОС, оскільки С++ — це чітко типізована та швидка мова програмування, що робить її ідеальним вибором для розробки операційної системи;

- ігри:

через те, що С++ є однією з найшвидших мов програмування, вона широко використовується в програмуванні двигунів розробки ігор. С++ може легко маніпулювати апаратними ресурсами, а також може забезпечувати процедурне програмування для інтенсивних функцій ЦП;

- браузери:

механізми візуалізації різних веб-браузерів запрограмовані на С++ завдяки швидкості, яку він пропонує;

- бібліотеки:

багато бібліотек високого рівня використовують С++ як основну мову програмування. Наприклад, кілька бібліотек машинного навчання використовують С++ у серверній частині через його швидкість;

- графіка:

С++ широко використовується майже в усіх графічних програмах, які вимагають швидкого рендерингу, обробки зображень, фізики в реальному часі та мобільних датчиків;

- банківські програми:

одна з найпопулярніших базових банківських систем – Infosys Finacle, використовує С++ як серверну мову програмування. Банківські додатки потребують щодня обробляти мільйони транзакцій і вимагають підтримки високого паралелізму та низької затримки;

- хмарні/розподілені системи:

хмарні системи зберігання даних використовують масштабовані файлові системи, які працюють близько до апаратного забезпечення. Ось чому С++ стає кращим вибором для хмарних систем;

- вбудовані системи:

різні вбудовані системи, такі як медичні машини, розумні годинники тощо, використовують C++ як основну мову програмування;

- компілятори:

компілятори різних мов програмування використовують C++ як базову мову програмування [56].

Для розробки проміжного програмного забезпечення для мультикомп'ютерної системи згідно сотової топології було обрано саме мову C++ через наступні її переваги:

- продуктивність: C++ відома своєю високою продуктивністю, що робить її ідеальним вибором для розробки програмного забезпечення для потужних систем. У мультикомп'ютерних системах, де можуть бути великі обсяги даних та потрібна швидка обробка, ефективність виконання коду є критично важливою. C++ дозволяє ближче підходити до апаратного рівня та максимізувати продуктивність;
- мультиплатформеність: C++ є мультиплатформеною мовою програмування, що означає, що написаний на ній код може працювати різних операційних системах і апаратних платформах без значних змін. Це особливо важливо в мультикомп'ютерних системах, де можуть використовуватися різні комп'ютери з різними конфігураціями та ОС. Можна розробляти програмне забезпечення на C++ і використовувати його на будь-якій платформі, що підтримується, що дозволяє забезпечити єдність системи;
- низькорівневий доступ до ресурсів: C++ надає низькорівневий доступ до апаратного забезпечення, наприклад: пам'ять, процесор, мережні інтерфейси та інші ресурси. Це дозволяє розробнику ефективно керувати ресурсами системи та оптимізувати їх використання. У мультикомп'ютерних системах, де важливо ефективно використовувати ресурси та забезпечувати швидку комунікацію між комп'ютерами, C++ дозволяє реалізувати високопродуктивні операції без зайвого тиражу;
- розширюваність: C++ підтримує об'єктно-орієнтоване програмування (ООП), що дозволяє створювати модульний код, який легко розширювати та підтримувати. У мультикомп'ютерних системах, де можуть бути різні компоненти

та модулі, важливо мати гнучку архітектуру, що дозволяє додавати та модифікувати функціональність без значних змін у вже існуючому коді;

- багата бібліотека: C++ має широкий вибір стандартних та сторонніх бібліотек, які допомагають прискорити розробку та спростити реалізацію різноманітних функцій. В мультикомп'ютерних системах можуть бути потрібні різні функціональні можливості, такі як мережеве взаємодія, обробка даних, криптографія тощо. З використанням наявних бібліотек можна значно зменшити зусилля, необхідні для розробки всіх цих функцій з нуля [57].

Також при написанні коду було використано сокети для забезпечення зв'язку між різними компонентами програми через мережу.

У програмуванні сокети є інтерфейсом (або API) для мережевого програмування. Вони дозволяють розробникам створювати програми, які можуть взаємодіяти з мережею і обмінюватися даними з іншими програмами, що працюють на віддалених комп'ютерах. За допомогою сокетів можна створювати клієнтські і серверні програми, реалізовувати різні мережеві протоколи і виконувати передачу даних через мережу.

У програмуванні сокети можуть використовуватись для роботи з різними протоколами, такими як TCP (Transmission Control Protocol) і UDP (User Datagram Protocol). TCP забезпечує надійне і з'єднане з'єднання, в той час як UDP працює в режимі без з'єднання і передає датаграми незалежно один від одного.

В загальному розумінні сокети представляють собою механізм для встановлення з'єднання та обміну даними між програмами через мережу. Вони надають програмам можливість взаємодіяти з мережевими протоколами і передавати дані між різними пристроями.

Сокети в програмуванні зазвичай використовуються для реалізації клієнт-серверних взаємодій. Клієнтська програма створює сокет і спробує підключитися до сервера за певною адресою і портом. Серверний бік, в свою чергу, слухає на певному порті і приймає вхідні з'єднання від клієнтів. Після встановлення з'єднання клієнт і сервер можуть обмінюватися даними через сокет, надсилаючи й отримуючи повідомлення.

В різних мовах програмування існують спеціальні бібліотеки або модулі для роботи з сокетами, які надають зручний інтерфейс для створення сокетів, з'єднання, передачі даних і обробки подій мережевого взаємодії.

Сокети дозволяють реалізовувати різні типи мережевих програм, таких як веб-сервери, чат-клієнти, файлообмінні додатки і багато інших. Вони є важливим інструментом для комунікації і обміну даними в мережевому середовищі [58].

3.2 Децентралізовані системи

Мультикомп'ютерна система згідно сотової топології, що була обрана для виконання даної дипломної роботи, є децентралізованою системою.

Децентралізація відома як розподіл функцій між кількома підрозділами. Це взаємопов'язана система, де жоден суб'єкт не має повних повноважень. Це архітектура, в якій робочі навантаження, як апаратне, так і програмне забезпечення, розподіляються між кількома робочими станціями.

Функції розподіляються між кількома машинами в децентралізованій системі замість того, щоб покладатися на один сервер. Вони мають кількох центральних власників. Власники можуть зберігати ресурси так, щоб кожен користувач мав доступ. Систему можна уявити графічно. Машину кожного користувача можна візуалізувати як вузли, які з'єднані один з одним. Кожен вузол має копію даних іншого вузла, а кілька власників також мають копії всіх вузлів, щоб зменшити час доступу. Тож щоразу, коли вноситься оновлення чи зміна даних вузла, ці зміни також відображаються в копіях. Проілюструємо на прикладах: Bitcoin — це новітня технологія та яскравий приклад децентралізованої системи. Це блокчейн, де немає центрального органу влади. Будь-хто може стати частиною мережі, брати участь у транзакціях і брати участь у голосуванні. Рішення приймається більшістю голосів. Dogecoin — це децентралізована однорангова криптовалюта, яка дозволяє нам здійснювати транзакції [59].

Треба мати чітку концепцію децентралізації, централізації та розподілених мереж (рис 3.1, рис 3.2, рис 3.3). У централізованій мережі є центральний мережевий орган, який приймає рішення. У децентралізованій системі є кілька

власників. Розподілені системи є подальшим розширенням децентралізації. Тут немає поняття власників. Усі користувачі є власниками та мають рівні права.

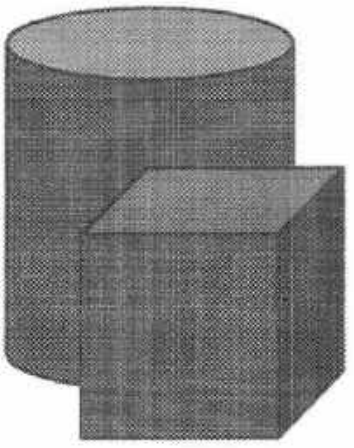


Рисунок 3.1 – Централізована система

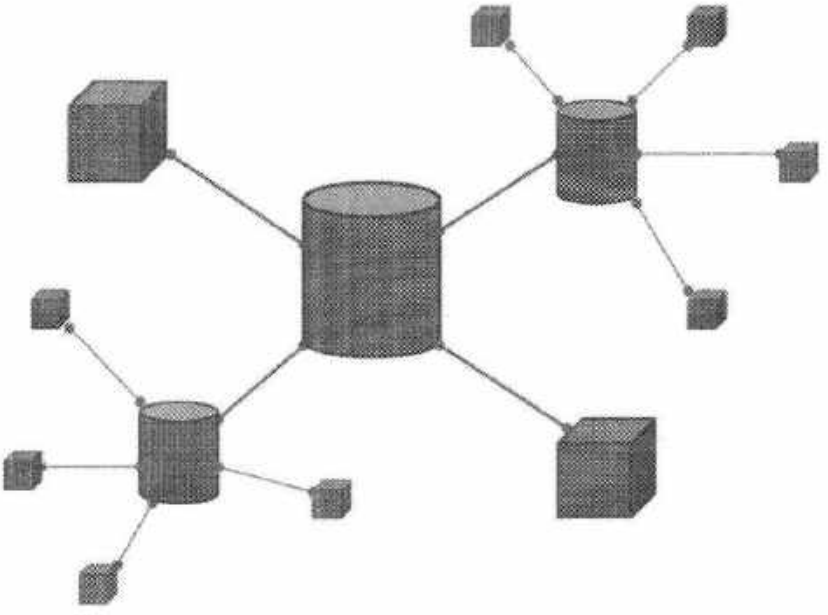


Рисунок 3.2 – Розподілена система

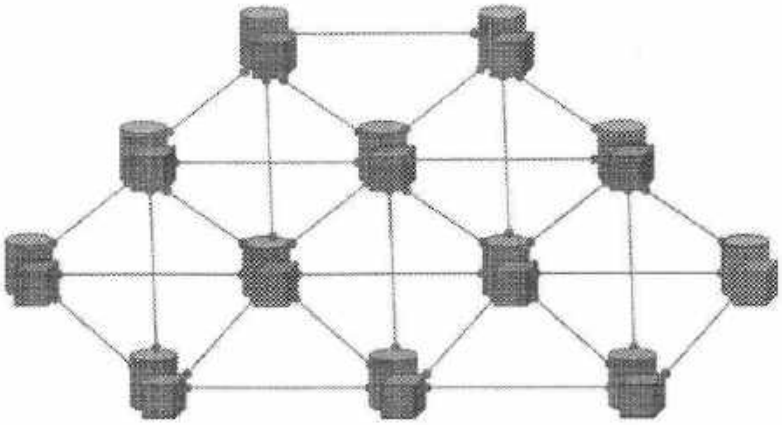


Рисунок 3.3 – Децентралізована система [60]

Важливість децентралізації. Децентралізація дуже важлива з наступних причин:

- оптимізація ресурсів: кожен користувач не обов'язково повинен мати всі ресурси. Децентралізоване налаштування дозволяє користувачеві розділити свій тягар з іншими на нижчому рівні;
- більший результат: оскільки всі користувачі мають однакові повноваження, кожен користувач працює з більшою ефективністю, щоб підвищити максимальну продуктивність;
- гнучкість: користувачі можуть ділитися власними поглядами, оскільки жодні центральні органи не накладають жодних обмежень. Вони також мають можливість змінювати свої рішення.

Робота децентралізації. У децентралізованій мережі користувач хоче поділитися деякими даними. Користувачеві не потрібно отримувати дозвіл, тому він має повне право публікувати щось. У мережі кожен користувач пов'язаний один з одним. Отже, коли користувач ділиться даними, вони передаються між іншими за допомогою протоколів. Користувачі схвалюють дані. Коли дані затверджені, протоколи оновлюють базу даних. Для відстеження всієї інформації ведеться база даних.

Необхідність децентралізації. У наш час технології прогресують з кожним днем, і кількість користувачів також збільшується. Для адміністрування системи централізована система не відповідає всім критеріям. Тож децентралізована система з кожним днем стає дуже корисною. Це створює ефективне, безпечне та надійне адміністрування. Це покращує однорангові мережі. Це забезпечує право кожного користувача. Кожна людина має право на прийняття рішень. Користувачі також мають доступ до бази даних. Найбільша перевага децентралізації полягає в тому, що якщо частина мережі виходить з ладу, вся мережа працюватиме без перебоїв. Основною причиною, чому децентралізація краща за централізацію, є гнучкість і дані для швидкої адаптації до вимог ринку.

Чи захищена децентралізована система? Децентралізація дуже захищена. Оскільки немає центрального органу, немає центрального сервера. Сервер кожного користувача діє як центральний сервер. Тому існує кілька серверів. Злом усіх серверів – неможливий варіант. Тому зараз багато організацій переходять на децентралізовану мережу. Яскравим прикладом є Google. Продукти Google від онлайн-пошуку до мобільного Android мають свободу працювати незалежно.

Основні переваги децентралізації:

- основна перевага полягає в тому, що в разі збою деяких вузлів або основного вузла вся система не руйнується;
- процес прийняття рішень здійснюється на основі голосування;
- зазвичай це відкриті платформи розробки, і цензури менше;
- до мережі можна додати більше машин.

Звісно, такі системи також мають низку своїх недоліків:

- шкалування: децентралізовані системи часто стикаються з проблемами шкалування. Збільшення кількості учасників і обсягу даних може призвести до перевантаження мережі та зниження її продуктивності. Розробка ефективних алгоритмів шкалування є важливим завданням у децентралізованих системах;
- ефективність: децентралізовані системи, особливо ті, що використовують блокчейн технологію, можуть бути менш ефективними з точки зору швидкості обробки та виконання операцій порівняно з централізованими

системами. Це може бути обмеженням для деяких додатків, особливо якщо вони потребують високої швидкості обробки;

- вартість: децентралізовані системи можуть вимагати значних витрат на утримання та операції. Наприклад, майнінг у блокчейні вимагає значних обчислювальних ресурсів та енергетичних витрат. Крім того, інфраструктура для децентралізованих систем може бути складною та дорогою для побудови та підтримки;
- безпека: хоч як було сказано вище, що децентралізовані системи пропонують більшу безпеку за рахунок розподіленості та криптографічних методів, вони також можуть бути вразливими до деяких видів атак. Наприклад, у децентралізованих фінансових системах можуть існувати ризики злочинної діяльності, такої як крадіжка криптовалюти або шахрайство;
- вирішення конфліктів: у децентралізованих системах прийняття рішень та вирішення конфліктів може бути складним завданням. Відсутність централізованої влади означає, що не завжди є чіткі механізми для вирішення суперечок або розбіжностей між учасниками системи;
- потреба в участі: децентралізовані системи можуть вимагати великої кількості учасників для забезпечення своєї ефективності та безпеки. Якщо система не має достатньої критичної маси учасників, вона може бути менш стійкою до атак та вразливостей.

Все ж таки, незважаючи на ці недоліки, децентралізація пропонує багато переваг, що забезпечують стабільну роботу системи, таких як більша приватність, відсутність цензури та контролю з боку централізованих організацій, а також більша стійкість до витоків даних та збоїв.

3.3 Архітектура програмного забезпечення

Код програми, написаний в ході виконання даної дипломної роботи, представляє проміжне програмне забезпечення, яке виконує обробку та пересилання даних між ноутбуками (laptops) та точками доступу (access points) через мережу.

Основний функціонал коду наступний:

- створення списків точок доступу та ноутбуків з відповідними IP-адресами та портами:

```
std::vector<std::pair<std::string, int>> access_points = {{"192.168.0.1", 5000},
                                                          {"192.168.0.2", 5000},
                                                          {"192.168.0.3", 5000}};
std::vector<std::pair<std::string, int>> laptops = {{"192.168.0.4", 5000},
                                                    {"192.168.0.5", 5000},
                                                    {"192.168.0.6", 5000}};
```

- створення сокету receive_socket для отримання даних від ноутбуків. Сокет прив'язується до адреси '0.0.0.0' і порту 5000. Цей сокет очікує на отримання даних від ноутбуків:

```
int receive_socket = socket(AF_INET, SOCK_DGRAM, 0);
sockaddr_in receive_address{};
receive_address.sin_family = AF_INET;
receive_address.sin_addr.s_addr = htonl(INADDR_ANY);
receive_address.sin_port = htons(5000);
bind(receive_socket, reinterpret_cast<struct sockaddr*>(&receive_address), sizeof(receive_address));
```

- у безкінечному циклі програма очікує отримання даних від ноутбуків за допомогою recvfrom(). Отримані дані виводяться на екран разом з адресою ноутбука, що їх надіслав:

```
ssize_t received_bytes = recvfrom(receive_socket, buffer, sizeof(buffer), 0,
                                  reinterpret_cast<struct sockaddr*>(&client_address),
                                  &client_address_length);
```

- після отримання даних від ноутбуків, програма пересилає ці дані до всіх точок доступу. Для кожної точки доступу зі списку access_points, створюється новий сокет send_socket, і дані надсилаються за допомогою sendto():

```
int send_socket = socket(AF_INET, SOCK_DGRAM, 0);
sendto(send_socket, data.c_str(), data.size(), 0, reinterpret_cast<struct sockaddr*>(&access_point_address), sizeof(access_point_address));
```

- після надсилання даних до точок доступу, програма очікує відповідей від кожної точки доступу. Для кожної точки доступу зі списку access_points, використовується той самий сокет receive_socket для отримання відповіді за допомогою recvfrom(). Отримані відповіді виводяться на екран разом з адресою точки доступу, що їх надіслала:

ssize_t received_bytes = recvfrom(receive_socket, response_buffer, sizeof(response_buffer), 0, nullptr, nullptr);

std::cout << "Received response from " << access_point.first << ": " << response << std::endl;

- після отримання відповідей від точок доступу, програма пересилає ці відповіді до відповідних ноутбуків. Для кожного ноутбука зі списку laptops, створюється новий сокет send_socket, і відповіді надсилаються за допомогою sendto();

int send_socket = socket(AF_INET, SOCK_DGRAM, 0);

sendto(send_socket, response.c_str(), response.size(), 0, reinterpret_cast<struct sockaddr*>(&laptop_address), sizeof(laptop_address));

- цикл повторюється, і програма продовжує очікувати та обробляти дані від ноутбуків, надсилати їх до точок доступу, отримувати відповіді від точок доступу та надсилати їх до ноутбуків;

- після завершення роботи програми, сокет receive_socket закривається за допомогою close().

close(receive_socket);

Отриманий функціонал дозволяє проміжному програмному забезпеченню отримувати дані від ноутбуків, пересилати їх до точок доступу, отримувати відповіді від точок доступу та пересилати їх до відповідних ноутбуків, створюючи зв'язок між ними через мережу.

У коді використано бібліотеку <iostream> для введення/виведення даних, <string> для роботи з рядками, <vector> для зберігання списків точок доступу та ноутбуків, <arpa/inet.h> для конвертації IP-адрес між різними форматами та <sys/socket.h> для роботи з сокетами.

Також, як було сказано у підрозділі вище, в програмі було використано технологію сокетів.

У цьому коді сокети використовуються для забезпечення зв'язку між різними компонентами програми через мережу. Основні функції сокетів в програмі такі:

- receive_socket - створюється сокет для отримання даних від ноутбуків. Він прив'язується до локальної адреси '0.0.0.0' і порту 5000. Цей сокет використовується для отримання даних, надісланих ноутбуками до проміжного програмного забезпечення;
- send_socket - створюється сокет для надсилання даних до точок доступу і ноутбуків. Цей сокет використовується для надсилання даних з проміжного програмного забезпечення до точок доступу та відповідей від точок доступу до відповідних ноутбуків;
- за допомогою функцій socket(), bind(), recvfrom() та sendto() з бібліотеки <sys/socket.h>, а також відповідних параметрів, сокети створюються, прив'язуються до певних адрес і портів, а також використовуються для отримання та надсилання даних через мережу.

Завершуючи роботу з сокетами, використовується функція close(), яка закриває сокети та звільняє пов'язані з ними ресурси.

Пояснення UML-діаграми:

- middleware() - головна функція, яка представляє проміжне програмне забезпечення. Вона містить список точок доступу (access_points) та ноутбуків (laptops), а також створює сокет для отримання даних від ноутбуків;
- receiveData() - клас, що відповідає за отримання даних від ноутбуків. Він має метод receiveData(), який отримує дані через сокет receive_socket;
- sendData() - клас, що відповідає за надсилання даних до точок доступу та ноутбуків. Він має метод sendData(), який надсилає дані до вказаної адреси та порту через сокет send_socket.

У діаграмі показано, що головна функція middleware() взаємодіє з класами receiveData() та sendData(), щоб отримувати дані від ноутбуків, надсилати їх до точок доступу та отримувати відповіді від точок доступу, які потім надсилаються до відповідних ноутбуків.

Нижче, на рисунку 3.2, показано UML діаграму згідно виконаної програми:

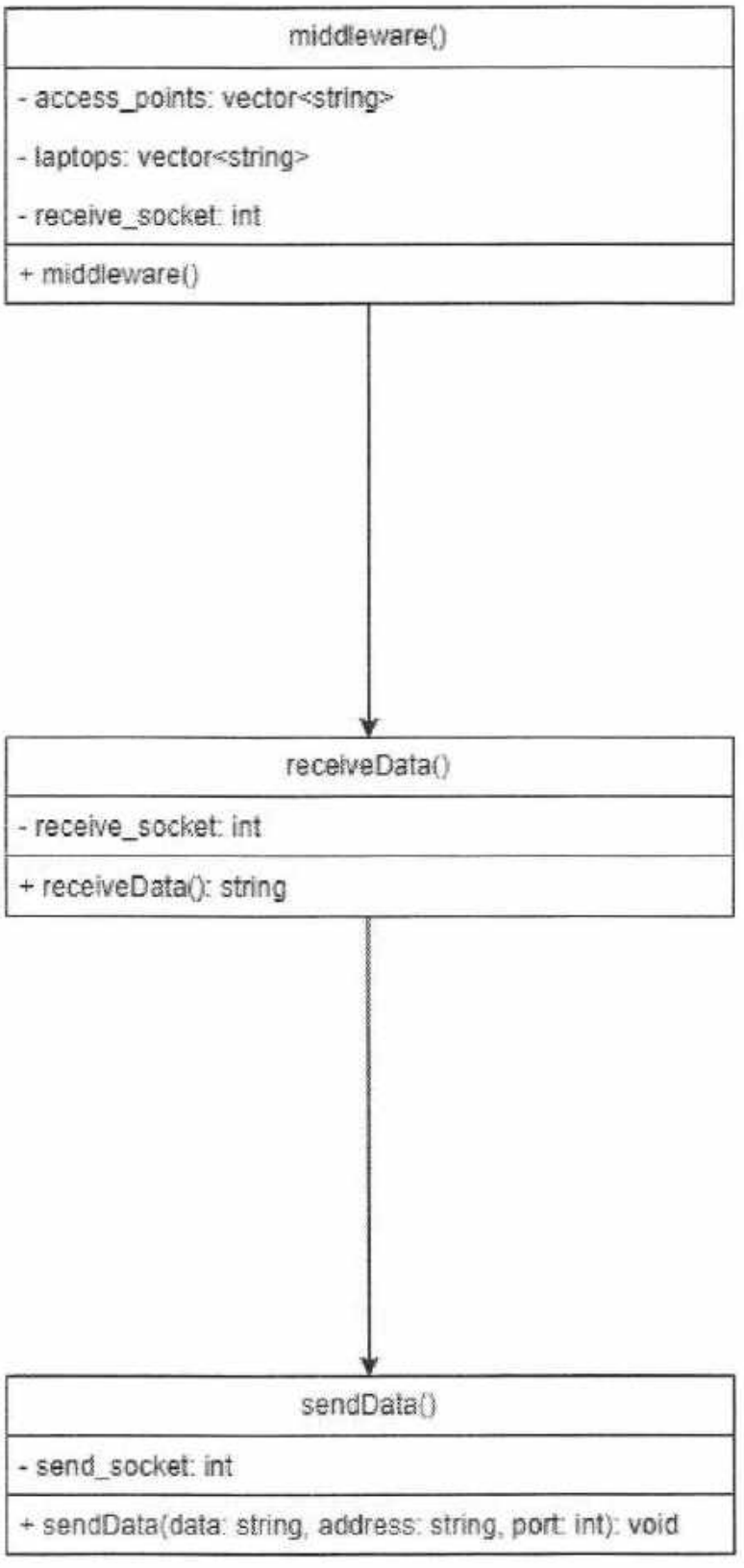


Рисунок 3.2 – UML-діаграма архітектури програмного забезпечення

3.4 Висновки

Отже, даний розділ дозволив провести аналіз і вибір необхідних програмних засобів, дослідити децентралізовані системи та визначити архітектуру програмного забезпечення, необхідну для успішної реалізації комп'ютерної системи.

Аналіз різних програмних засобів показав, що мова програмування C++ є ефективним вибором для реалізації системи, оскільки вона надає широкі можливості для розробки високопродуктивного та надійного програмного забезпечення.

Дослідження децентралізованих систем показало, що такий підхід може бути вигідним для створення комп'ютерних систем з більшою надійністю та масштабованістю. Використання розподіленого підходу дозволяє знизити ризик відмови окремих компонентів і забезпечити безперервну роботу системи.

Архітектура програмного забезпечення, обрана для комп'ютерної системи, відіграє важливу роль у забезпеченні її функціональності, ефективності та масштабованості. Виявлено, що обрана архітектура має бути гнучкою, забезпечувати розділення відповідальностей між компонентами системи та забезпечувати зручну розробку, тестування та супроводження програмного забезпечення.

Загальний висновок з розділу "Архітектура програмного забезпечення комп'ютерної системи" полягає у визнанні важливості правильного вибору програмних засобів, розробки децентралізованої системи та обрання архітектури, що відповідає вимогам функціональності та продуктивності. Ці кроки є критичними для успішної реалізації комп'ютерної системи і забезпечення її ефективної роботи.

ВИСНОВКИ

Отже, в дипломній роботі на тему "Мультикомп'ютерна система згідно сотової топології" було спроектовано та змодельовано дану систему, сновними елементами якої були 3 точки доступу, 3 ноутбуки та написане проміжне програмне забезпечення на мові C++ з використанням сокетів.

Метою дослідження було створення мультикомп'ютерної системи, яка базується на сотовій топології, що дозволяє забезпечити ефективний обмін даними між різними комп'ютерами у мережі. Для досягнення цієї мети були використані 3 точки доступу, які були розташовані у різних частинах мережі. Кожна точка доступу була підключена до ноутбука, який виконував функції клієнта.

Програмне забезпечення було розроблено на мові програмування C++ та включало в себе реалізацію сокетів - механізму, який дозволяє здійснювати обмін даними між комп'ютерами через мережу. Застосування сокетів дозволило забезпечити зручний та ефективний спосіб комунікації між компонентами системи.

Отже, ця дипломна робота дозволила розробити мультикомп'ютерну систему згідно сотової топології та показала її працездатність та ефективність. Результати дослідження можуть бути використані для подальшого розвитку подібних систем та вдосконалення процесу комунікації між комп'ютерами у мережі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Електронні обчислювальні машини. URL: https://sites.google.com/site/istoriaobcisluvanoietehniki/mehanicni-obcisluvalni-pristroie (дата звернення: 17.03.2023).

2. Електронна обчислювальна машина. URL: https://wiki.cuspu.edu.ua/index.php/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D0%B0_%D0%BE%D0%B1%D1%87%D0%B8%D1%81%D0%BB%D1%8E%D0%B2%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD%D0%B0 (дата звернення: 17.03.2023).

3. Chao Hong. Parallel Processing in Power Systems Computation on a Distributed Memory Message Passing Multicomputer. 2017. 186 p.

4. Saad Bani Mohammad. Efficient Processor Allocation Strategies for Mesh Multicomputers: Performance Evaluation of Processor Allocation Strategies for Mesh Interconnection Networks. 2009. 208 p.

5. РОЗПОДІЛЕНЕ СЕРЕДОВИЩЕ. URL: https://moodle.znu.edu.ua/pluginfile.php/486887/mod_resource/content/1/%D0%93%D0%BB%D0%BE%D0%B1%D0%B0%20%D0%BA%D0%BD%D0%B8%D0%B3%D0%B0%20%D0%A2%D0%BE%D0%BC1-39-52.pdf (дата звернення: 18.03.2023).

6. Рольщиков В. Б. Технології розподілених систем та паралельних обчислень. 2016. 155 с.

7. Архітектура паралельних обчислювальних систем. URL: https://uk.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BF%D0%B0%D1%80%D0%B0%D0%BB%D0%B5%D0%BB%D1%8C%D0%BD%D0%B8%D1%85_%D0%BE%D0%B1%D1%87%D0%B8%D1%81%D0%BB%D1%8E%D0%B2%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D1%85_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC (дата звернення: 18.03.2023).

8. Kevin C O'Kane. Basic IBM Mainframe Assembly Language Programming. 2011. 296 p.

9. Stephen H. Kaisler. Mainframe Computer Systems. 2020. 457 p.

10. Dinesh Dattani. IBM Mainframe Security: Beyond the Basics—A Practical Guide from a z/OS and RACF Perspective (Ebl-Schweitzer). 2013. 224 p.

11. James W. Cortada. IBM: The Rise and Fall and Reinvention of a Global Icon (History of Computing). 2019. 732 p.

12. What is IBM Z? URL: https://www.developer.com/cloud/what-is-ibm-z/ (дата звернення: 20.03.2023).

13. 6 Advantages and Disadvantages of Mainframe Computer | Drawbacks & Benefits of Mainframe Computer. URL: https://www.hitechwhizz.com/2021/02/6-advantages-and-disadvantages-drawbacks-benefits-of-mainframe-computer.html (дата звернення: 20.03.2023).

14. Мережні операційні системи. URL: https://studfile.net/preview/7134850/ (дата звернення: 23.03.2023).

15. Класифікація та особливості операційних систем. URL: https://pidru4niki.com/19710518/menedzhment/klasifikatsiya_osoblivosti_operatsiynih_sistem (дата звернення: 23.03.2023).

16. Призначення та основні функції ОС. URL: https://operatsijna-sistema.webnode.com.ua/priznachennya-ta-osnovni-funktsiji-os/ (дата звернення: 23.03.2023).

17. Структура ОС. URL: https://operatsijna-sistema.webnode.com.ua/struktura-os/ (дата звернення: 23.03.2023).

18. Операційні системи. URL: https://dut.edu.ua/ua/news-1-626-9874-operaciyni-sistemi_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy (дата звернення: 23.03.2023).

19. Що таке Linux? URL: http://ipkey.com.ua/uk/faq/924-linux.html (дата звернення: 23.03.2023).

20. MAC OS. URL: https://sites.google.com/site/systemoshkaoperativnaya/mac-os (дата звернення: 23.03.2023).

21. Еволюція мобільної операційної системи від Apple - від iPhone OS до iOS 8. URL: https://blog.allo.ua/evolyutsiya-mobilnoj-operatsionnoj-sistemy-ot-apple-ot-iphone-os-do-ios-8_2015-04-12/ (дата звернення: 23.03.2023).

22. Однозадачні і багатозадачні ОС. Приклади. Переваги багатозадачності. URL: http://um.co.ua/3/3-17/3-170690.html (дата звернення: 25.03.2023).

23. Різниця між операційною системою однокористувацької та багатокористувацької. URL: https://uk.strephonsays.com/difference-between-single-user-and-multiuser-operating-system (дата звернення: 25.03.2023).

24. Симетричні і асиметричні мультипроцесорні системи. URL: http://um.co.ua/3/3-15/3-157277.html (дата звернення: 25.03.2023).

25. Системи реального часу. URL: https://pidru4niki.com/1050030361395/informatika/sistemi_realnogo_chasu (дата звернення: 25.03.2023).

26. Мережеві операційні системи. URL: https://pidru4niki.com/10810806/bankivska_sprava/merezhevi_operatsiyni_sistemi (дата звернення: 25.03.2023).

27. What Is Middleware? URL: https://aws.amazon.com/what-is/middleware/?nc1=h_ls (дата звернення: 02.04.2023).

28. Middleware: What it is and How Does it Work? URL: https://geekflare.com/middleware-work/ (дата звернення: 02.04.2023).

29. Проміжне програмне забезпечення, орієнтоване на повідомлення. URL: https://hmn.wiki/uk/Message-oriented_middleware (дата звернення: 02.04.2023).

30. Проміжні програмні засоби. URL: https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BC%D1%96%D0%B6%D0%BD%D1%96_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D1%96_%D0%B7%D0%B0%D1%81%D0%BE%D0%B1%D0%B8 (дата звернення: 02.04.2023).

31. Топологія комп'ютерних мереж. URL: https://stud.com.ua/53329/informatika/topologiya_kompyuternih_merezh (дата звернення: 06.04.2023).

32. Engr. Wilson Kurt. Network Topology: The Physical and Logical Structure of a Network connection Between Model and nodes. 2019. 59 p.

33. Gerardus Blokdyk. Network Topology A Complete Guide - 2020 Edition. 2020. 304 p.

34. Computer Networks. URL: http://www.fimt-ggsipu.org/study/bca210.pdf (дата звернення: 09.04.2023).

35. Топологія зірка. URL: https://kovelpost.com/blogs/213 (дата звернення: 09.04.2023).

36. Топологія типу «Дерево». URL: https://studfile.net/preview/5152835/page:6/ (дата звернення: 09.04.2023).

37. What is a Topology? URL: https://fcit.usf.edu/network/chap5/chap5.htm (дата звернення: 09.04.2023).

38. Computer Network Design for Universities in Developing Countries. URL: https://www.semanticscholar.org/paper/Computer-Network-Design-for-Universities-in-AlSarhan/1438d602d647d363d309b9797cb5072579a66add (дата звернення: 09.04.2023).

39. Computer Networking concepts. URL: http://mucins.weebly.com/22-topology.html (дата звернення: 09.04.2023).

40. Paul Bedell. Cellular Networks: Design and Operation - A Real World Perspective. 2014. 408 p.

41. Adrián Cardalda García, Stefan Maier, Abhay Phillips. Location-Based Services in Cellular Networks: from GSM to 5G NR. 2020. 480 p.

42. Бойко М. П. Системи стільникового зв'язку. 2004. 70 с.

43. Стільниковий зв'язок. URL: https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D1%96%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA%D0%BE%D0%B2%D0%B8%D0%B9_%D0%B7%D0%B2'%D1%8F%D0%B7%D0%BE%D0%BA (дата звернення: 13.04.2023).

44. Ноутбук Lenovo IdeaPad 1 15IGL7 (82V7003WRA). URL: https://rozetka.com.ua/ua/370038426/p370038426/ (дата звернення: 16.04.2023).

45. Точка доступу Ubiquiti NanoStation Loco M5. URL: https://defis.ua/tochka-dostupu-ubiquiti-nanostation-loco-m5.html (дата звернення: 16.04.2023).

46. Точка доступу Ubiquiti Rocket 5ac (R5AC-Lite). URL: https://rozetka.com.ua/ua/339040336/p339040336/ (дата звернення: 16.04.2023).

47. Антена Wi-Fi Ubiquiti AirMax Omni 5G (AMO-5G10). URL: https://rozetka.com.ua/ua/239853511/p239853511/ (дата звернення: 16.04.2023).

48. Extending home wifi using Ubiquiti nanobeam M2 radio. URL: https://community.ui.com/questions/Need-help-extending-home-wifi-using-Ubiquiti-nanobeam-M2-radio/8ae848a0-6f79-421d-acf3-3047b19b1f07 (дата звернення: 18.04.2023).

49. Огляд веб-інтерфейсу Ubiquiti Nanostation M5 (AirOS v5.2). URL: https://www.technotrade.com.ua/Articles/ubiquiti-nanostation-m5-webinterface.php?comments_page=2 (дата звернення: 18.04.2023).

50. Огляд і настройка Ubiquiti Nanostation M2. URL: https://www.ateasyday.com/articles/devices/obzor-i-nastrojka-ubiquiti-nanostation-m2.html (дата звернення: 18.04.2023).

51. Ремонт Ubiquiti за рахунок відновлення прошивки. Як повернути Compliance Test Ubnt. URL: https://ntools.com.ua/uk/information/faq/remont-ubiquiti-za-schet-vosstanovlenija-proshivki (дата звернення: 18.04.2023).

52. Вмикаємо airMAX в Ubiquiti 802.11a/b/g обладнання. URL: https://www.technotrade.com.ua/Articles/airmax_in_ubiquiti_11abg.php (дата звернення: 18.04.2023).

53. Nanostation M5 Problem configuration Access Point (Hostpot). URL: https://community.ui.com/questions/Nanostation-M5-Problem-configuration-Access-Point-Hostpot/8b9da743-c11c-4b3a-a84b-39c90eb33ad5 (дата звернення: 18.04.2023).

54. Система контролю за зависанням мережевого обладнання. URL: https://lanmarket.ua/ua/stats/sistema-kontrolya-za-zavisaniem-setevogo-oborudovaniya-ili-nastraivaem-storojevoy-taymer/ (дата звернення: 18.04.2023).

55. Підключення домашніх Wi-Fi пристроїв до Ubiquiti M серії. URL: https://www.technotrade.com.ua/Articles/ubiquiti_m_compatibility_settings.php?comments_page=3 (дата звернення: 18.04.2023).

56. What is C++? Basic Concepts of C++ Programming Language. URL: https://www.guru99.com/cpp-tutorial.html (дата звернення: 20.04.2023).

57. A Comprehensive Guide to C++: Advantages and Disadvantages. URL: https://www.pangea.ai/dev-web-development-resources/a-comprehensive-guide-to-c-advantages-and-disadvantages/ (дата звернення: 20.04.2023).

58. Lewis Van Winkle. Hands-On Network Programming with C: Learn socket programming in C and write secure and optimized network code. 2019. 478 p.

59. Pavel Kravchenko, Bohdan Skriabin, Oksana Dubinina. Blockchain And Decentralized Systems. 2019. 446 p.

60. Are distributed networks and decentralized systems the same? URL: https://hub.packtpub.com/are-distributed-networks-decentralized-systems-same/ (дата звернення: 23.04.2023).

Додаток А

(обов'язковий)

Лістинг коду проміжного програмного забезпечення

```
#include <iostream>
#include <string>
#include <vector>
#include <arpa/inet.h>
#include <sys/socket.h>
void middleware() {
  // IP-адреси точок доступу та порти
  std::vector<std::pair<std::string, int>> access_points = {{"192.168.0.1",
5000},
                                                {"192.168.0.2", 5000},
                                                {"192.168.0.3", 5000}};
  std::vector<std::pair<std::string, int>> laptops = {{"192.168.0.4", 5000},
                                                {"192.168.0.5", 5000},
                                                {"192.168.0.6", 5000}};
  // Створення сокету для отримання даних від ноутбуків
  int receive_socket = socket(AF_INET, SOCK_DGRAM, 0);
  sockaddr_in receive_address{};
  receive_address.sin_family = AF_INET;
  receive_address.sin_addr.s_addr = htonl(INADDR_ANY);
  receive_address.sin_port = htons(5000);
  bind(receive_socket,                    reinterpret_cast<struct
sockaddr*>(&receive_address), sizeof(receive_address));
  while (true) {
    char buffer[1024];
    sockaddr_in client_address{};
    socklen_t client_address_length = sizeof(client_address);
```

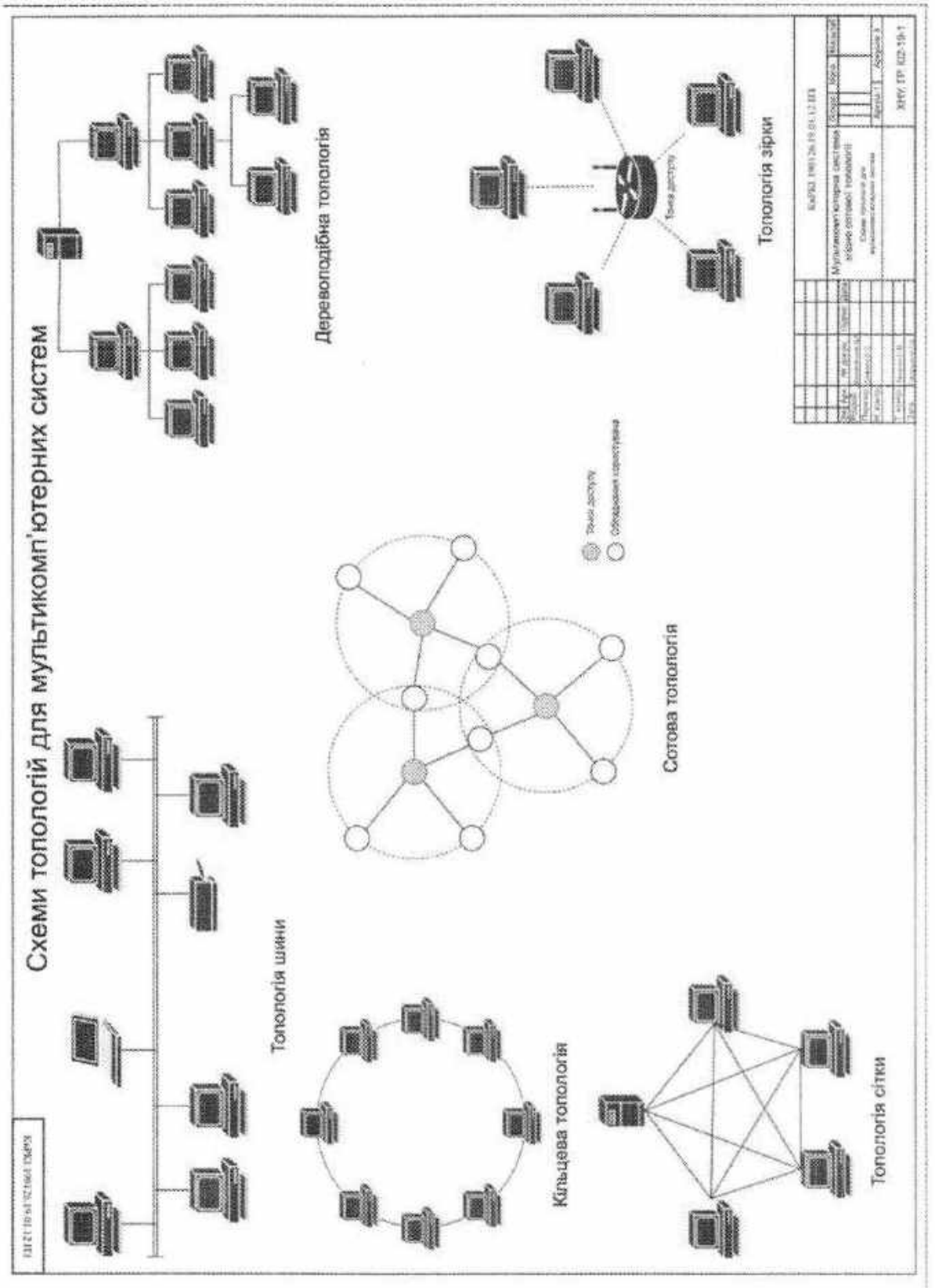
```
            ssize_t received_bytes = recvfrom(receive_socket, buffer,
sizeof(buffer), 0, reinterpret_cast<struct sockaddr*>(&client_address),
&client_address_length);
            if (received_bytes == -1) {
                std::cerr << "Failed to receive data from laptops." << std::endl;
                continue;
            }
            std::string data(buffer, received_bytes);
            std::cout << "Received data from " <<
inet_ntoa(client_address.sin_addr) << ": " << data << std::endl;
            // Пересилання даних до всіх точок доступу
            for (const auto& access_point : access_points) {
                int send_socket = socket(AF_INET, SOCK_DGRAM, 0);
                sockaddr_in access_point_address{};
                access_point_address.sin_family = AF_INET;
                access_point_address.sin_addr.s_addr =
inet_addr(access_point.first.c_str());
                access_point_address.sin_port = htons(access_point.second);
                sendto(send_socket, data.c_str(), data.size(), 0,
reinterpret_cast<struct sockaddr*>(&access_point_address),
sizeof(access_point_address));
                close(send_socket);
            }
            // Прийом відповідей від точок доступу
            for (const auto& access_point : access_points) {
                char response_buffer[1024];
                ssize_t received_bytes = recvfrom(receive_socket, response_buffer,
sizeof(response_buffer), 0, nullptr, nullptr);

                if (received_bytes == -1) {
```

```
                std::cerr << "Failed to receive response from access point " <<
access_point.first << std::endl;
                continue;
            }
            std::string response(response_buffer, received_bytes);
            std::cout << "Received response from " << access_point.first << ": "
<< response << std::endl;
            // Пересилання відповідей до відповідних ноутбуків
            for (const auto& laptop : laptops) {
                int send_socket = socket(AF_INET, SOCK_DGRAM, 0);
                sockaddr_in laptop_address{};
                laptop_address.sin_family = AF_INET;
                laptop_address.sin_addr.s_addr = inet_addr(laptop.first.c_str());
                laptop_address.sin_port = htons(laptop.second);
                sendto(send_socket, response.c_str(), response.size(), 0,
reinterpret_cast<struct sockaddr*>(&laptop_address), sizeof(laptop_address));
                close(send_socket);
            }
        }
    }
    close(receive_socket);
}

int main() {
    // Запуск проміжного програмного забезпечення
    middleware();

    return 0;
}
```

Додаток Б

(обов'язковий)

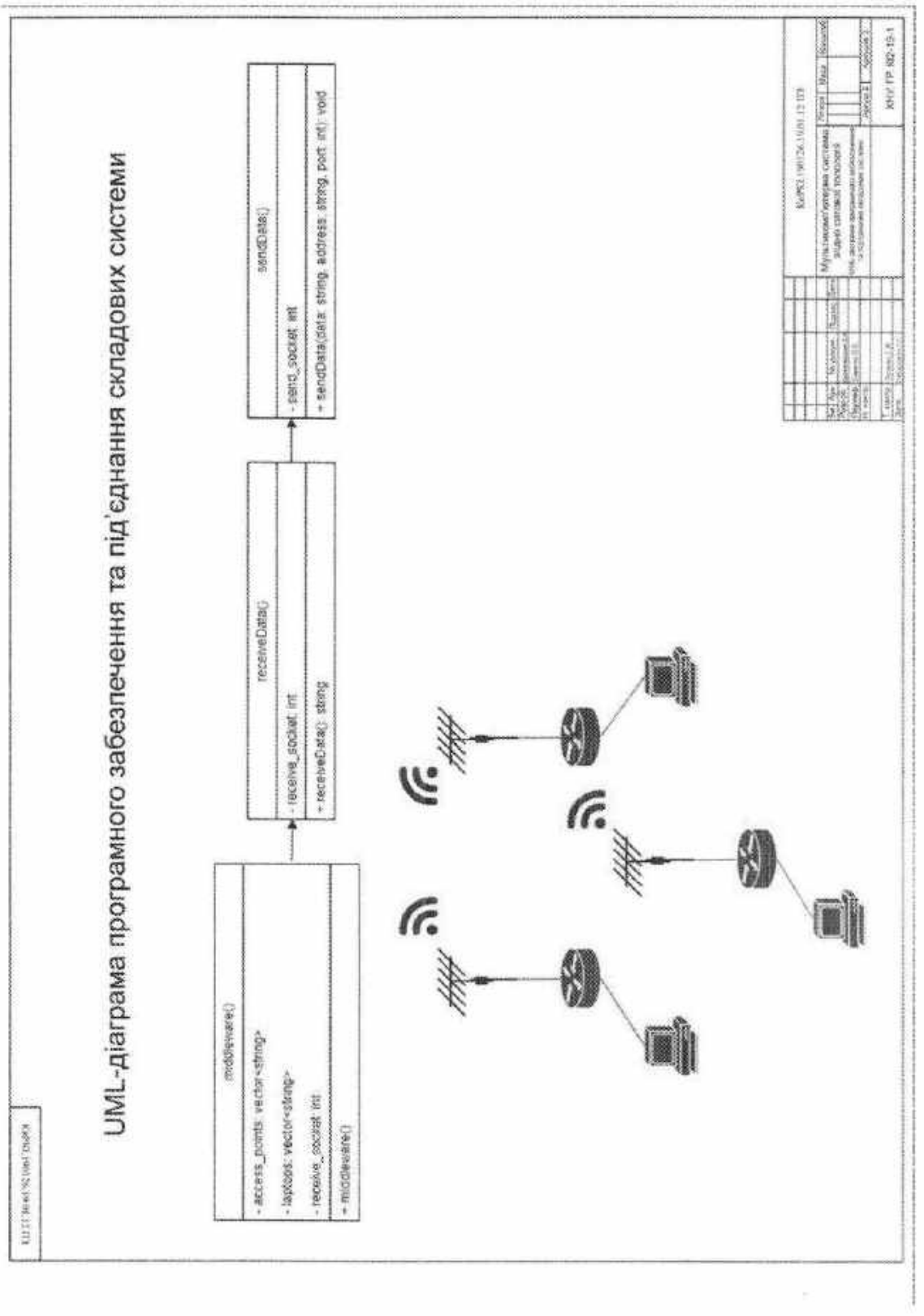
Копія креслення «Схеми топологій для мультикомп'ютерних систем»



Додаток В

(обов'язковий)

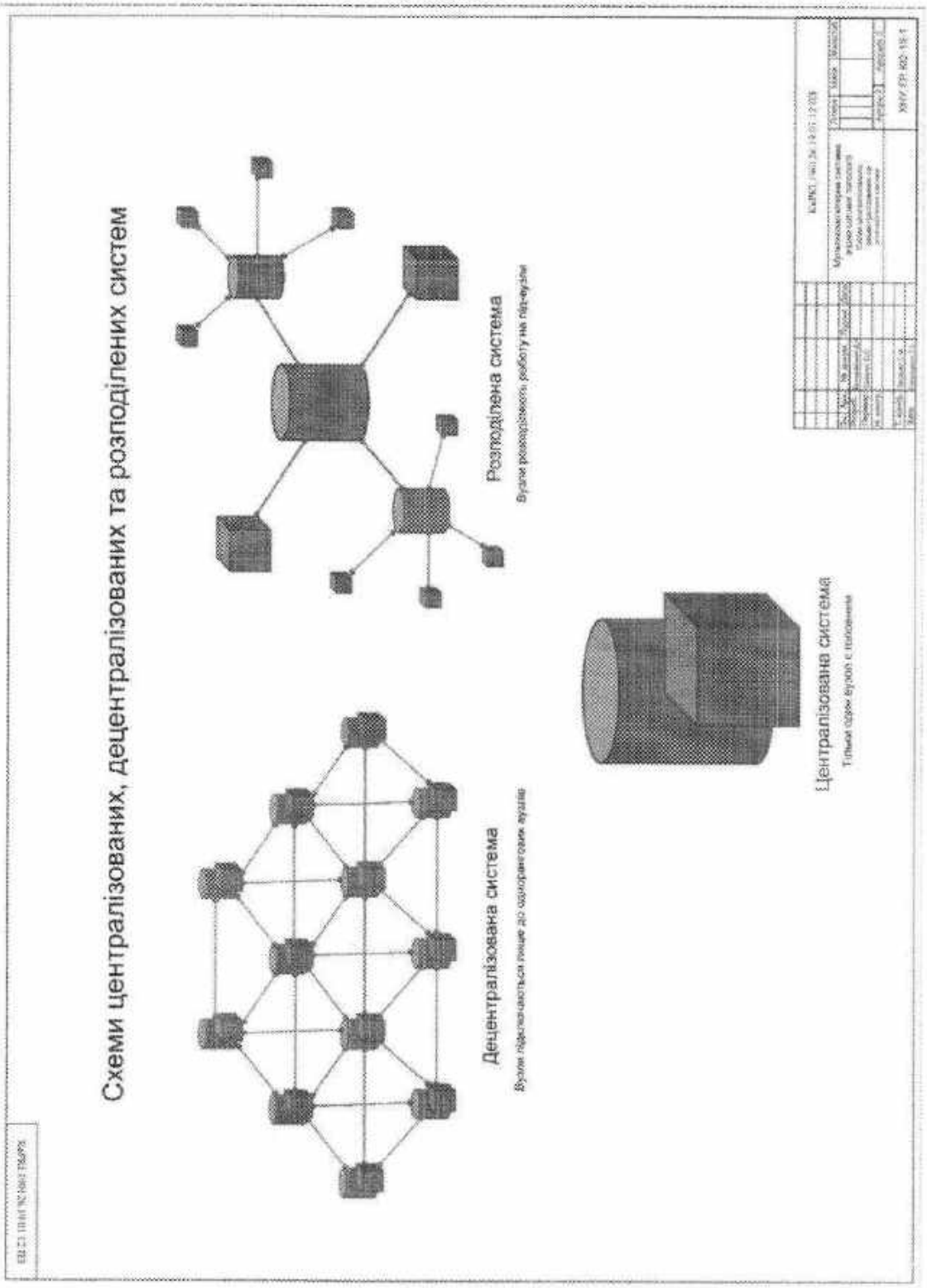
Копія креслення «UML-діаграма програмного забезпечення та під'єднання складових системи»



Додаток Г

(обов'язковий)

Копія креслення «Схеми централізованих, децентралізованих та розподілених систем»



UNICHECK

Ім'я користувача:
Кафедра КІ

ID перевірки:
1015378603

Дата перевірки:
01.06.2023 21:39:55 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
01.06.2023 21:41:03 EEST

ID користувача:
100005591

Назва документа: **Крижанівський_Мультикомп'ютерна система згідно сотової топології**

Кількість сторінок: **66** Кількість слів: **11266** Кількість символів: **88886** Розмір файлу: **2.82 MB** ID файлу: **1015044087**

14.6%
Схожість

Найбільша схожість: **5.31%** з Інтернет-джерелом (https://techukraine.net/%D1%89%D0%BE-%D1%86%D0%B5-%D1%82%...

14.4% Джерела з Інтернету | 600 ..... Сторінка 68

5.25% Джерела з Бібліотеки | 127 ..... Сторінка 75

0.07% Цитат

Цитати | 1 ..... Сторінка 76

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 13.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 12%**

ID: 114510 Назва: БКР Мультикомп'ютерна система згідно сотової топології Додано в БД: 2023-06-01 Автора: Д.А. Крижанівський Керівники: О.С. Савенко Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	76161	658	11573 (15%)	94 (14%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Крижанівський Дмитро Андрійович

Тема: Мультикомп'ютерна система згідно сотової топології

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 Кількість сторінок записки 55

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є проектування та моделювання мультикомп'ютерної системи згідно сотової топології

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 та поставлено задачу (проаналізовано поняття мультикомп'ютерної системи, розглянуто найбільш підходяще апаратне забезпечення для виконання задачі, а також проаналізовано наявне системне та програмне забезпечення). В другому розділі було розглянуто всі найчастіше використовувані топології систем, описано їх особливості та перераховано переваги і недоліки, обгрунтовано вибір саме сотової топології, детально розписано її технологію та структуру, а також затверджено перелік апаратного забезпечення для ефективної реалізації мультикомп'ютерної системи, було обрано три ноутбуки у якості користувацького обладнання та 3 точки доступу, які будуть моделювати роботу стільникового зв'язку. В третьому розділі кваліфікаційної роботи було проведено аналіз програмних засобів, зроблено вибір у сторону мови C++ та змодельовано роботу мультикомп'ютерної системи завдяки проміжному програмному забезпеченню, що дозволяє обмінюватись даними між користувацьким обладнання та точками доступу.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: отримані результати потребують різних варіантів моделювання.

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: відмінно

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Шертинов Валерій Володимирович, д.т.н., професор, зав. каф. АКП ХНУ

"5" 06 2023 р. _____ (підпис)

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

Крижанівського Дмитра Андрійовича
ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-19-1

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.

05.06.2023

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: Мультикомп'ютерна система згідно сотової топології

Автор: Крижанівський Дмитро Андрійович

Спеціальність: 123 – Компютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Березька Катерина Миколаївна, к.т.н, доцент

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 доопрацьована 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, про що свідчить посилання системи на збіг з 10-30 джерелами на один фрагмент речення.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ідентичності/схожості, складає 14.6% і адресується до 777 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ К. М. Березька

Гарант ОП _____ С. М. Лисенко

Завідувач кафедри КПС _____ Т. О. Говорущенко