Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: «Розробка графічного симулятора для візуалізації та дослідження 7-рівневої моделі OSI і стеку протоколів TCP/IP з використанням фреймворку JavaFX»

Виконав: студент 4 курсу

групи 5ПІ-21б

спеціальності

121 – Інженерія програмного забезпечення

(шифр і назва напряму підготовки, спеціальності)

Сулим М. Ю.

(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Катєльніков Д. І.

(прізвище та ініціали)

Рецензент: к.т.н., доц. каф.ОТ Городецька О.С.

(прізвище та ініціали)

Допущено до захисту

Зав. кафедри ПЗ Романюк О. Н.

«09» 06 2025 р.

ВНТУ – 2025

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
О. Н. Романюк
« 21 » березня 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Сулим Мирославу Юрійовичу

1. Тема роботи – розробка графічного симулятора для візуалізації та дослідження 7-рівневої моделі OSI і стеку протоколів TCP/IP з використанням фреймворку JavaFX.

Керівник роботи: Катєльніков Денис Іванович, к. т. н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. №97.

2. Строк подання студентом роботи 2 червня 2025.

3. Вхідні дані – конфігурації пристроїв, координати об'єктів, параметри інтерфейсів, запити користувача. Вихідні дані – візуалізація мережі та пакетів, обробка подій, зміни стану мережі.

4. Вступ; обґрунтування актуальності теми та вибору напряму розробки симулятора; аналіз моделей OSI та TCP/IP, а також способів їх подання у візуальному середовищі; обґрунтування архітектури симулятора та вибору інструментів розробки; побудова логіки взаємодії мережевих пристроїв; реалізація модулів симуляції, візуалізації, конфігурації, збереження стану, теоретичного блоку; тестування функціональності симулятора; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу: аналоги; діаграми класів модулів застосунку; графічна схема застосунку; блок-схеми алгоритмів роботи застосунку; графічний інтерфейс застосунку.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-4	Катєльніков Д. І., к.т.н, доцент кафедри ПЗ	24.03.25	01.06.25

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз моделей мережевої взаємодії та вибір напрямів досліджень	25.03.25 – 08.04.25	вик
2	Розробка моделі та алгоритмів графічного симулятора	09.04.25 – 16.04.25	вик
3	Розробка модулів реалізації симулятора	17.04.25 – 03.05.25	вик
4	Тестування програмного застосунку	04.05.25 – 19.05.25	вик
5	Оформлення матеріалів до захисту БКР	20.05.25 – 30.05.25	вик

Студент (підпис) М. Ю. Сул (ініціали та прізви

Керівник бакалаврської кваліфікаційної роботи (підпис) Д. І. Катєльнік (ініціали та прізви

АНОТАЦІЯ

УДК 004.912.032.26

Сулим М.Ю. Розробка програмного симулятора для візуалізації та дослідження 7-рівневої моделі OSI та стеку протоколів TCP/IP з використанням фреймворку JavaFX: бакалаврська кваліфікаційна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2025. 105 с.

На укр. мові. Бібліогр.: 32 назв; рис.: 48 ; табл.: 1.

У бакалаврській кваліфікаційній роботі розроблено програмний застосунок для візуалізації та дослідження 7-рівневої моделі OSI та стеку протоколів TCP/IP. Основна увага приділена побудові інтуїтивно зрозумілого симуляційного середовища з інтерактивною візуалізацією мережевих процесів. Застосунок реалізовано з використанням мови програмування Java та фреймворку JavaFX для створення графічного інтерфейсу користувача.

Розроблено модулі симуляції передачі пакетів, маршрутизації, обробки протоколів. Впроваджено модуль подій, для відтворення поведінки мережі в реальному часі, та інтерфейси для конфігурації параметрів пристроїв. Також приділено увагу можливості отримання теорії безпосередньо в межах симулятора.

Отриманий в результаті застосунок може бути використаний у навчальних цілях або як основа для створення складніших симуляційних середовищ.

Ключові слова: комп’ютерна мережа, симулятор, OSI, TCP/IP, JavaFX.

ABSTRACT

Sulym M.Y. Development of a software simulator for visualization and research of the 7-layer OSI model and the TCP/IP protocol stack. Vinnytsia: VNTU, 2025. 105 p.

In Ukrainian language. Bibliography: 32 titles; fig.: 48; table.1.

In the bachelor's qualification work, a software application has been developed for visualization and research of the 7-layer OSI model and the TCP/IP protocol stack. The main attention is paid to building an intuitive simulation environment with interactive visualization of network processes. The application is implemented using the Java programming language and the JavaFX framework to create a graphical user interface.

Modules for simulating packet transmission, routing, and protocol processing have been developed. An event module has been implemented to reproduce the network behavior in real time, and interfaces for configuring device parameters. Attention has also been paid to the possibility of obtaining theory directly within the simulator.

The resulting application can be used for educational purposes or as a basis for creating more complex simulation environments.

Keywords: computer network, simulator, OSI, TCP/IP, JavaFX.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У сучасному інформаційному суспільстві комп’ютерні мережі [1] відіграють ключову роль у забезпеченні обміну даними між людьми, пристроями, сервісами та інформаційними системами [2]. Більшість сфер антропогенної діяльності покладаються на можливості, які надають мережеві технології. Щодня, велика кількість користувачів взаємодіють із нею, навіть не замислюючись як функціонує комп’ютерна мережа. У зв'язку з цим існує потреба створення засобів для вивчення її основних принципів роботи та підвищення обізнаності при використанні або підтримці комп’ютерних систем.

При вивченні багаторівневих мережевих технологій [3], концепцій моделі OSI [4] та стеку протоколів TCP/IP [5] виникає необхідність у вивченні теоретичних аспектів і практичних підходів для організації обміну даними, структуризації трафіку, маршрутизації, передачі інформації та взаємодії між пристроями. Для опанування цієї області знань недостатньо лише текстових описів чи статичних схем – необхідна візуалізація та моделювання.

Зокрема, навчання базових принципів функціонування 7-рівневої моделі OSI [4] та стеку протоколів TCP/IP ускладнене високим рівнем абстракції [6], але користувачам необхідно розуміти, як інформація переходить з одного пристрою до іншого, як вона змінюється на кожному етапі обробки, які структури даних при цьому використовуються та як відбувається координація подій між різними компонентами мережі. Теоретичне засвоєння таких понять без практичного закріплення часто призводить до фрагментарного або формального знання [7], що не дозволяє зрозуміти загальну картину функціонування інформаційної взаємодії.

Вивчення сфери комп’ютерних мереж напряму залежить від використаних освітніх засобів. У цьому контексті велику роль відіграють симулятори – програмні засоби, які дозволяють в інтерактивному режимі спостерігати, моделювати й аналізувати роботу мережі в умовно реальному часі [8].

Симулятор, орієнтований на навчання, дає змогу не лише відтворювати процес передачі даних, але й наочно демонструвати, як інформація проходить через різні рівні логічної моделі, як пристрої взаємодіють між собою та як виникають або опрацьовуються певні мережеві ситуації.

У порівнянні з іншими навчальними засобами інтерактивний графічний симулятор для візуалізації та дослідження семирівневої моделі OSI та стеку протоколів TCP/IP використовує сучасні технології комп'ютерної графіки, що дозволяє значно покращити засвоєння матеріалу, а отже, робить його розробку актуальним завданням.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою бакалаврської кваліфікаційної роботи є розширення функціоналу навчальних засобів для дисципліни «Організація комп'ютерних мереж» та посилити наочність навчального матеріалу за рахунок створення програмних засобів графічного симулятора для візуалізації та дослідження функціонування 7-рівневої моделі OSI та стеку протоколів TCP/IP.

Відповідно до поставленої мети у кваліфікаційній роботі необхідно вирішити такі завдання:

– проаналізувати засоби та методи для реалізації симулятора мережевих технологій;

– розробити архітектуру та модель роботи симулятора;

– створити алгоритми взаємодії рівнів моделі OSI та TCP/IP;

– розробити графічний інтерфейс застосунку;

– реалізувати модульну структуру симулятора, що забезпечить інтерактивність і наочність навчання;

– провести тестування й налагодження розробленого програмного продукту.

Об’єкт дослідження – процес розробки графічного симулятора для візуалізації та дослідження роботи 7-рівневої моделі OSI та стеку протоколів TCP/IP.

Предмет дослідження – методи і засоби для розробки графічного симулятора.

Методи дослідження. У процесі дослідження використовувалися такі методи:

- теорія комп'ютерних мереж та методи системного аналізу для вивчення структури процесів у комп'ютерних мережах;

- методи програмної інженерії, зокрема моделювання архітектури програмного забезпечення, побудова користувацьких інтерфейсів і логіки взаємодії компонентів системи;

- комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Новизна отриманих результатів.

1. Подальшого розвитку отримав метод візуалізації та симуляції процесів взаємодії рівнів моделі OSI та стеку TCP/IP, який, на відміну від існуючих, використовує абстрактні класи та рівні наслідування, що дозволяє інтерпретувати мережеві процеси у наочній формі з можливістю інтерактивної зміни параметрів і спостереження результатів у режимі реального часу.

2. Подальший розвиток отримали метод побудови мережевої топології, який, на відміну від існуючих, має вбудований чат для використання штучного інтелекту [10], що дозволяє вчасно отримувати коментарі щодо правильного виконання операцій з'єднання мережевих компонент.

Практична цінність отриманих результатів. Практична цінність полягає в тому, що на основі отриманих в бакалаврській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмний засіб графічного симулятора для візуалізації та дослідження 7-рівневої моделі OSI і стеку протоколів TCP/IP.

Особистий внесок здобувача. Всі результати, отримані в рамках бакалаврської кваліфікаційної роботи, досягнуті самостійно автором.

У друкованій праці [9], опублікованій у співавторстві, автору належать такі результати: дослідження механізмів Properties & Bindings, розробка архітектури програмного засобу та створення подійно-орієнтованої схеми обробки вхідних даних. У друкованій праці [11], опублікованій у співавторстві, автору належать такі результати: розробки програмного забезпечення від формулювання ідеї до реалізації, тестування та оформлення результатів.

Апробація матеріалів кваліфікаційної роботи. Основні положення бакалаврської кваліфікаційної роботи доповідалися та обговорювалися на міжнародних та всеукраїнських конференціях: LIV Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (ВНТУ, 2025) [11], I Міжнародній науково-практичній конференції «Innovations in Science: From Theoretical Foundations to Practical Impact» (Антверпен, 2025) [9].

Публікації. Основні результати дослідження опубліковано в 2 наукових працях, які були доповіддю на зазначених конференціях та опубліковані в відповідних матеріалах.

1 АНАЛІЗ РОЗВИТКУ ЗАСОБІВ МОДЕЛЮВАННЯ МЕРЕЖЕВИХ ТЕХНОЛОГІЙ ТА ПОСТАНОВКА ЗАДАЧ РОЗРОБКИ

1.1 Аналіз стану симуляторів моделювання мережевих технологій

У сучасних умовах, коли цифрові технології проникають в усі сфери життя, важливим елементом підготовки фахівців у галузі інформаційних технологій є глибоке розуміння принципів функціонування комп'ютерних мереж. Передавання даних у мережевих інфраструктурах підпорядковується суворим правилам, описаним у багаторівневих моделях [3], зокрема моделі OSI та стеку TCP/IP. Навчання цим моделям потребує не лише теоретичного вивчення, а й практичного закріплення знань через моделювання процесів обміну пакетами між пристроями.

Однак поточний стан використання симуляторів комп'ютерних мереж в освітньому процесі засвідчує певні недоліки та обмеження. Передусім, значна частина навчального процесу обмежується поясненням принципів взаємодії між пристроями, без візуального підтвердження або відтворення реальних мережевих сценаріїв. Це ускладнює сприйняття матеріалу користувачами, особливо у випадках, коли йдеться про логіку роботи транспортного або мережевого рівня [4] – наприклад, визначення маршруту, адресацію чи трансляцію доменних імен [12].

Серед основних викликів, які супроводжують навчання мережевим технологіям, можна виділити відсутність інструментів для поетапної демонстрації передачі пакетів, відправки запитів, маршрутизації та обробки відповідей. Часто навчальний процес обмежується спрощеними моделями або лише текстовим описом, що не дає можливості повноцінно простежити, як пакет проходить усі рівні моделі OSI – від прикладного до канального.

Ще одним недоліком є недостатній рівень інтерактивності [13] існуючих підходів до вивчення мереж. Користувач не завжди має змогу самостійно змінювати параметри пристроїв, експериментувати з IP-адресацією [14],

створювати конфліктні або діагностичні ситуації, щоб зрозуміти, як відбувається обробка помилок або пошук альтернативного шляху.

Також варто зазначити обмежені можливості візуалізації самого процесу передачі даних. Багато наявних програмних застосунків не дозволяють бачити, як саме пакет переміщується мережею – які пристрої задіяні, яка логіка використовується для вибору маршруту, коли та чому виникає запит певного типу. Це ускладнює розуміння складних сценаріїв, де бере участь декілька рівнів моделі та взаємодіє багато пристроїв.

Таким чином, поточний стан реалізацій моделювання комп'ютерних мереж свідчить про потребу у створенні нового підходу, який поєднує доступність, гнучкість, візуальність і можливість взаємодії з усіма ключовими рівнями OSI-моделі. Інструмент, який дозволяє змінювати налаштування, бачити результат у реальному часі, досліджувати структуру мережевого стеку та відтворювати логіку протоколів – є не лише бажаним, а й необхідним етапом у модернізації освітнього процесу в галузі комп'ютерних мереж.

1.2 Порівняльний аналіз аналогів

Моделювання комп'ютерних мереж є невід'ємною частиною сучасної IT-освіти, оскільки дозволяє користувачам практично ознайомитись із принципами роботи мережевих протоколів, передачею даних та налаштуванням пристроїв. З розвитком цифрових технологій з'явився ряд програмних засобів, які реалізують можливості симуляції мережевої взаємодії. Серед найбільш відомих продуктів, що широко використовуються в освітньому та професійному середовищі, можна виділити:

– Cisco Packet Tracer;

– Graphical Network Simulator 3;

– Emulated Virtual Environment Next Generation;

– Boson NetSim.

Cisco Packet Tracer є одним із найвідоміших і найпоширеніших симуляторів комп'ютерних мереж, розробленим корпорацією Cisco спеціально для студентів освітньої платформи Cisco Networking Academy [15]. Додаток надає зручне графічне середовище для створення топологій мереж, налаштування пристроїв, симуляції маршрутизаторів, комутаторів, серверів і клієнтів. Він підтримує основні протоколи, зокрема, а також деякі функції на рівні канального та транспортного рівнів OSI.

Основною перевагою Packet Tracer є його простота в опануванні, що робить його зручним для новачків та учнів, які тільки починають вивчати комп'ютерні мережі. Програмне забезпечення має інтуїтивний інтерфейс, зручну навігацію (див. рисунок 1.1), просту логіку налаштування пристроїв та візуальне відображення базових процесів передачі даних.

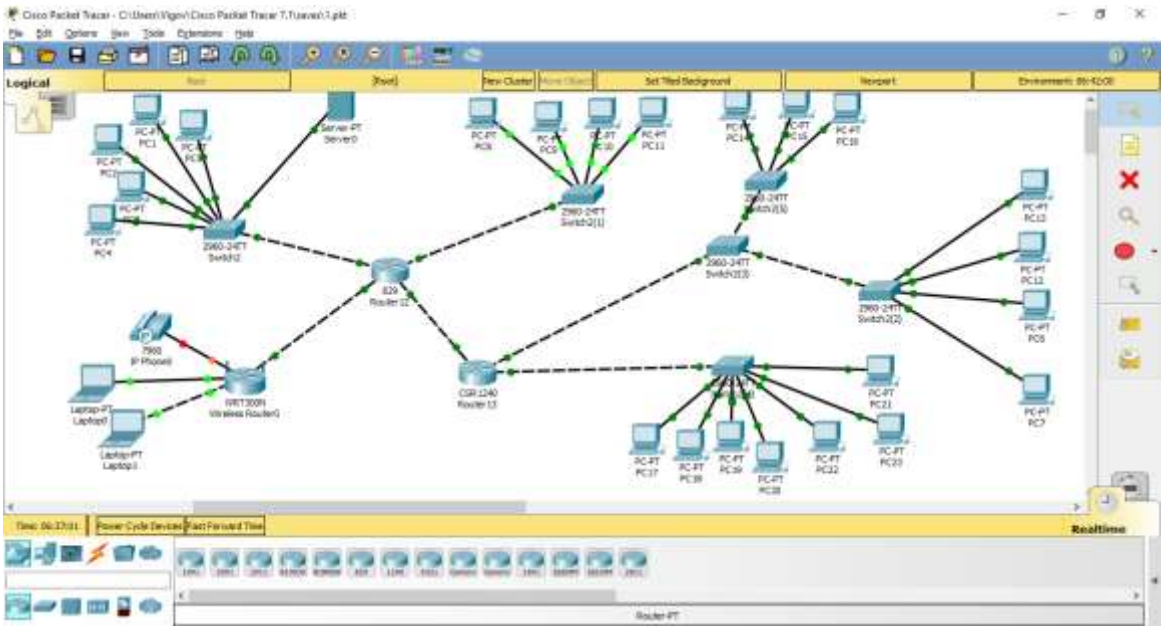


Рисунок 1.1 – Інтерфейс застосунку Cisco Packet Tracer

Проте Packet Tracer має й суттєві обмеження. Зокрема, він є закритим за архітектурою – користувач не має змоги змінювати внутрішню логіку реалізації протоколів або адаптувати систему під нові сценарії. Багато процесів працюють «під капотом», без змоги дослідити, як саме пакет обробляється на кожному рівні OSI-моделі. Також немає глибокої деталізації руху пакету – від прикладного

рівня до фізичного. Це знижує його ефективність у глибокому вивченні протоколів або при моделюванні нетипових ситуацій.

Graphical Network Simulator 3 – це професійний симулятор, орієнтований на побудову реалістичних мережевих лабораторій шляхом використання образів операційних систем реальних мережевих пристроїв [16]. Graphical Network Simulator 3 активно використовується мережевими інженерами для підготовки до сертифікацій, тестування мережевих конфігурацій і дослідження складних мережевих сценаріїв. Симулятор дозволяє інтегрувати віртуальні машини, запускати реальні маршрутизатори та комутатори, створювати мережі різного масштабу.

Головна перевага Graphical Network Simulator 3 – реалістичність, користувач працює не з віртуальною моделлю, а з повноцінною емуляцією операційної системи пристрою (див. рисунок 1.2). Це дозволяє відтворити поведінку мережевих систем максимально близько до справжньої.

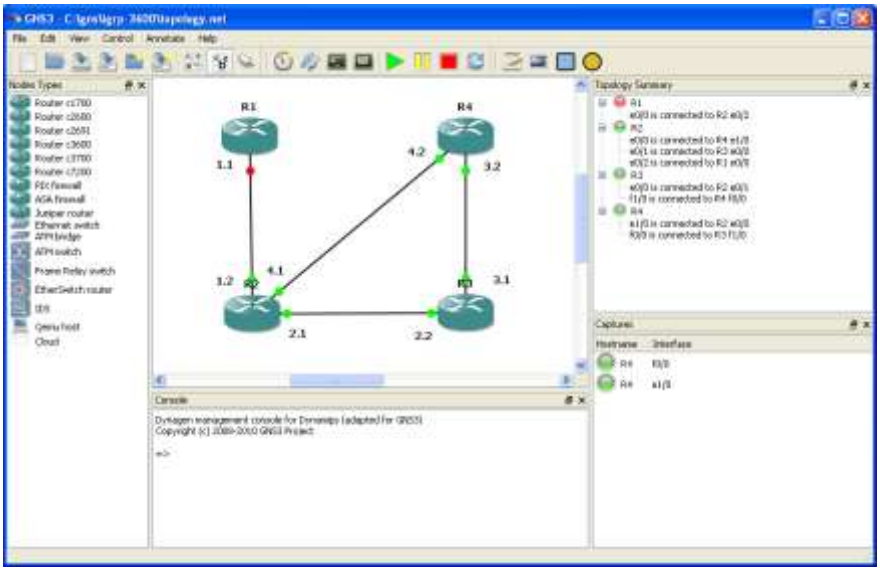


Рисунок 1.2 – Інтерфейс застосунку Graphical Network Simulator 3

Водночас Graphical Network Simulator 3 має високий поріг входу. Для його використання необхідні значні апаратні ресурси, глибокі знання мережевих протоколів, операційних систем, термінального середовища. Крім того, інтерфейс не підтримує інтерактивну візуалізацію переміщення пакетів –

користувач не бачить, як саме проходить пакет мережею, не може простежити етапи обробки. Це робить інструмент малопридатним для освітніх потреб початкового рівня.

Emulated Virtual Environment Next Generation – ще один емулятор мережевого середовища, побудований на основі використання віртуалізації та образів мережевих операційних систем [17]. Як і Graphical Network Simulator 3, він дозволяє створювати повноцінні мережі з реальними пристроями та тестувати складні конфігурації для корпоративного чи датацентрового середовища.

Програмний застосунок підтримує створення хмарної інфраструктури, багаторівневих віртуальної приватної мережі, віртуальної локальної мережі, тунелювання, сценарії з динамічними протоколами маршрутизації. Однак, як і Graphical Network Simulator 3, вона має орієнтацію на професійний рівень, що обумовлює складність інтерфейсу (див. рисунок 1.3), необхідність попереднього розгортання та конфігурування серверного середовища. Відсутня візуалізація логіки обробки пакетів: користувач не бачить розкладання пакету на заголовки, не може спостерігати за роботою окремих рівнів OSI.

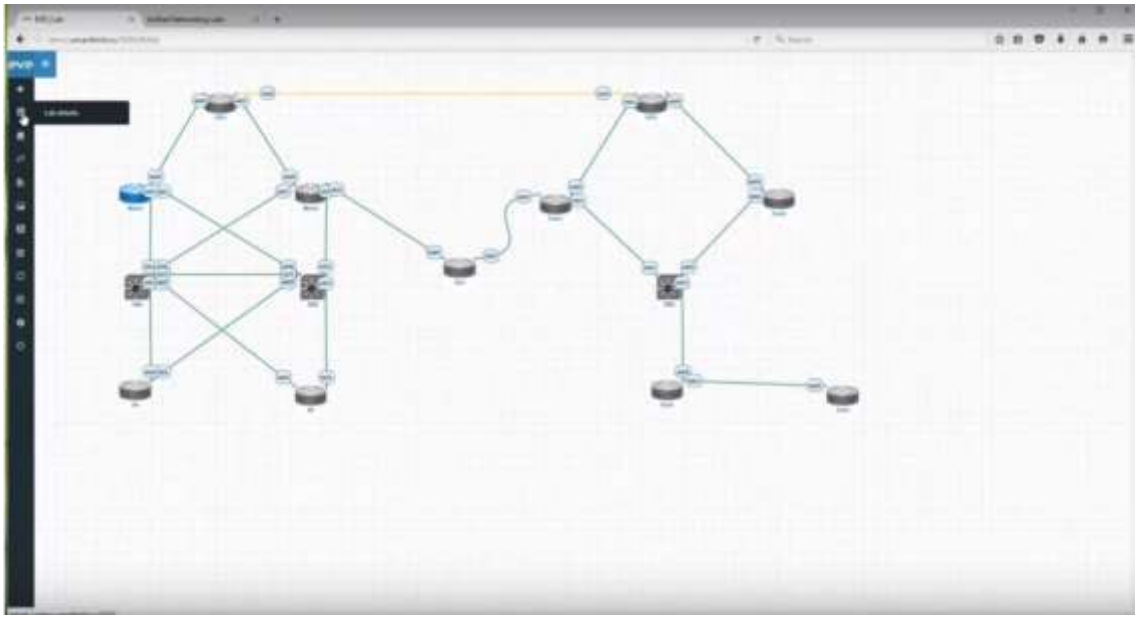


Рисунок 1.3 – Інтерфейс застосунку Emulated Virtual Environment Next Generation

Boson NetSim – це комерційне програмне забезпечення, спеціально розроблене для підготовки до сертифікаційних іспитів [18]. Воно містить сотні інтерактивних сценаріїв, які охоплюють налаштування мережевих пристроїв, діагностику та практичне завдання з протоколами.

Boson NetSim орієнтований на відтворення реальних командних інтерфейсів, тому він є корисним інструментом для тих, хто готується до іспитів. Основна перевага – велика база лабораторних робіт із поясненням, що дозволяє відпрацювати практичні навички (див. рисунок 1.4).

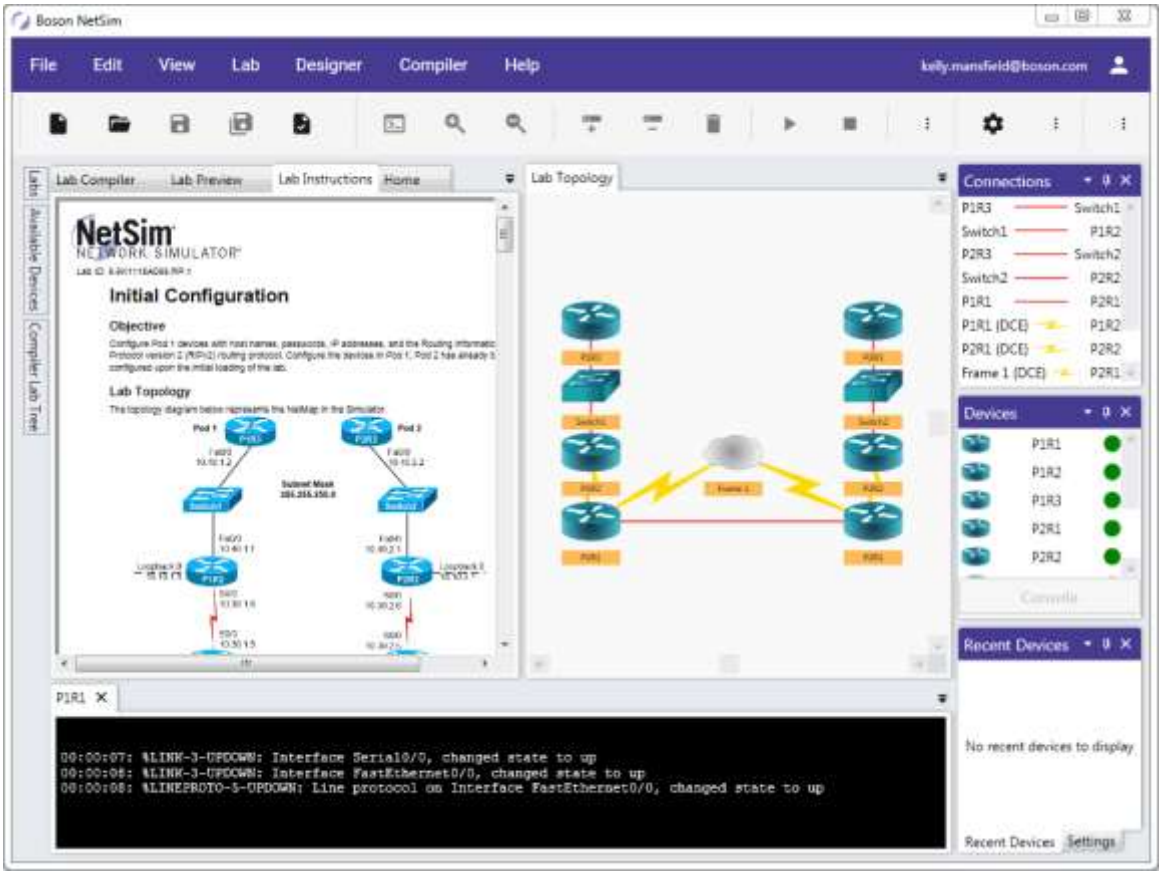


Рисунок 1.4 – Інтерфейс застосунку Boson NetSim

Однак реалізація закрита для модифікацій. Вона не передбачає зміни сценаріїв, створення нестандартних ситуацій, або додавання нових типів протоколів. Також, незважаючи на наявність текстових логів, відсутня візуалізація передачі пакетів або робота з рівнями OSI. Інтерфейс переважно командний, що ускладнює процес навчання для візуалів або початківців.

Розробка симулятора комп’ютерних мереж передбачає створення універсального та комплексного підходу, яке забезпечує широкі можливості для моделювання мережевих процесів з урахуванням потреб освітнього середовища та дослідницьких завдань. Програмне середовище має підтримувати гнучке налаштування мережевої інфраструктури, включаючи створення топологій, конфігурацію пристроїв та візуалізацію обміну даними. Ефективна взаємодія між елементами симуляції досягається завдяки впровадженню модулів, що відповідають за передачу пакетів, обробку протоколів різних рівнів та реєстрацію подій у мережі. Підвищення наочності навчального процесу забезпечується анімацією руху пакетів, виведенням стану інтерфейсів, таблиць маршрутизації та інших структур даних. Додатковою перевагою є система збереження проєктів, яка дозволяє користувачеві зберігати, відновлювати та аналізувати створені мережеві схеми, що розширює функціональність симулятора в межах академічного, практичного або дослідницького використання.

Результати порівняльного аналізу розглянутих аналогів та власної реалізації симулятора, під назвою Network Simulator наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерій	Packet Tracer	GNS3	EVE-NG	NetSim	Network Simulator
Підтримка базової мережевої логіки	+	+	+	+	+
Симуляція повноцінної передачі даних	+	+	-	-	+
Конфігурація пристроїв у реальному часі	+	-	+	+	+
Відкритість для розширення та модифікації	-	+	+	-	+
Використання гарячих клавіш	+	-	+	+	+

Продовження таблиці 1.1

Критерій	Packet Tracer	GNS3	EVE-NG	NetSim	Network Simulator
Теоретичний модуль	+	-	-	+	+
Загальна оцінка	83%	50%	66%	66%	100%

Беручи до уваги результати порівняльного аналізу, розробка власного симулятора комп'ютерних мереж є обґрунтованою. Запропонований симулятор поєднує переваги відомих реалізацій, доповнюючи їх функціональні можливості тими компонентами, які залишаються нереалізованими або обмеженими в існуючих програмних продуктах. Завдяки комплексному підходу, симулятор забезпечує глибоку візуалізацію мережевої взаємодії, підтримку ключових протоколів та відкритість до модифікацій, що робить його особливо цінним для навчальних та дослідницьких цілей.

Отже, створення власного програмного засобу для моделювання мережевих процесів є доцільним кроком у напрямку вдосконалення інструментів візуалізації та опису процесів мережевих технологій та експериментального аналізу мереж. Симулятор сприятиме підвищенню обсягу засвоєного матеріалу завдяки теоретичній, практичній і візуалізаційній складових, забезпечить більшу гнучкість для налаштувань та аналізу, а також відкриє нові можливості для інтеграції додаткових протоколів і сценаріїв. У підсумку, це програмне забезпечення здатне стати повноцінною альтернативою існуючим продуктам та сприяти розвитку навичок у галузі мережевих технологій.

1.3 Аналіз методів розв'язання задачі

Розробка програмного симулятора для моделювання мережевої взаємодії відповідно до 7-рівневої моделі OSI та стеку TCP/IP потребує ретельного вибору архітектурних концепцій і способів реалізації ключових функцій системи. Задача передбачає створення застосунку, що дозволятиме будувати мережеву

топологію, конфігурувати пристрої, моделювати передачу даних та наочно візуалізувати роботу базових протоколів.

Одним з основних питань при проєктуванні архітектури симулятора є вибір між монолітною та мікросервісною структурою. Мікросервісна архітектура дозволяє декомпонувати програмний застосунок на незалежні сервіси, кожен з яких відповідає за окрему частину логіки [19]. Це забезпечує масштабованість і гнучкість, однак у випадку симулятора, що має бути переважно локальним, навчальним застосунком, такий підхід створює зайву складність: потрібна організація міжсервісної взаємодії, синхронізація даних, розгортання кожного модуля окремо, що не відповідає формату невеликого десктопного застосунку.

Альтернативою є монолітний підхід, який передбачає побудову цілісного застосунку з розмежуванням функціональних компонентів всередині структури коду [19]. Цей варіант є простішим у реалізації, зручним у тестуванні й ефективним для локального середовища. Він дозволяє централізовано керувати логікою передачі пакетів, зберігати єдиний стан моделі, реалізовувати подієву обробку мережевих дій у межах одного застосунку. Саме тому на доцільно зупинитися саме на монолітній архітектурі, з можливістю подальшої декомпозиції, якщо проєкт буде розширюватись.

Для забезпечення зрозумілої та масштабованої реалізації симулятора доцільно використовувати багаторівневу архітектуру, яка дозволяє логічно розділити призначення між окремими компонентами програми. Перший рівень – графічний інтерфейс, що забезпечує взаємодію користувача з програмою: побудову мережевої топології, керування пристроями, перегляд інформації та конфігурування параметрів. Другий рівень – симуляційна логіка, яка відповідає за обробку подій, передачу мережевих пакетів між пристроями, реалізацію протоколів та управління чергою подій. Третій рівень це мережеві пристрої – представлений у вигляді об’єктів, які містять інтерфейси з параметрами та іншу локальну логіку. Четвертий рівень – збереження та завантаження проєктів – реалізує механізми серіалізації стану симуляції, що дозволяє зберігати створені

користувачем конфігурації мереж і надалі з ними працювати. А також додатковий модуль для теоретичної інформації з україно-англомовним довідником та чатом для використання штучного інтелекту Такий структурний підхід сприяє модульності, спрощує розробку і забезпечує гнучкість подальшого розширення функціоналу.

У якості базової концепції обробки подій доцільно застосувати об’єктно-орієнтовану архітектуру. Такий підхід дозволить симулювати не лише передачу пакетів, а й усі супутні процеси. Для цього можна створити окрему чергу подій, яка буде покроково або циклічно обробляти події у графічному симуляторі.

Важливим питанням також є реалізація протоколів. Для кожного з них планується створити окремі типи пакетів та логіку їх обробки в межах класів пристроїв. Це забезпечить модульність і розділення відповідальності між компонентами. Взаємодія між пристроями здійснюватиметься через інтерфейси з відповідними параметрами.

Таким чином, для розробки симулятора мережевої взаємодії пропонується реалізація монолітної багаторівневої архітектури, побудованої за принципами об’єктно-орієнтованого програмування, з гнучкою структурою класів для пристроїв, інтерфейсів і пакетів. Такий підхід дозволяє не лише реалізувати необхідний функціонал, а й забезпечити простоту тестування, розширення та візуалізації роботи мережі.

1.4 Постановка задач для розробки проєкту

На основі проведеного аналізу існуючих симуляторів, а також з урахуванням цілей та вимог до створення навчального програмного забезпечення для моделювання мережевої взаємодії, було сформульовано перелік задач, які необхідно реалізувати в межах проєкту:

– розробити модуль симуляції для обробки подій, передачі мережевих пакетів і логіки роботи протоколів згідно з моделлю OSI;

– розробити модуль побудови мережевої топології, що забезпечуватиме створення пристроїв, з’єднань між інтерфейсами та їхнє розміщення у візуальному середовищі;

– розробити модуль конфігурації пристроїв, який надаватиме можливість задавати параметри для відповідних пристроїв;

– розробити модуль збереження та завантаження проєктів, що реалізує серіалізацію всіх параметрів мережі та симуляції і дозволить відновити повний стан симуляції;

– розробити модуль для отримання супровідної теоретичної інформації;

– розробити графічний інтерфейс програмного застосунку, який забезпечуватиме взаємодію користувача з симулятором, включаючи інструменти додавання пристроїв, налаштування, запуску симуляції;

– провести тестування програмної системи на предмет відповідності поставленим вимогам.

1.5 Висновки

Отже, було з’ясовано, що сучасні інструменти моделювання комп’ютерних мереж не повністю задовольняють потреби користувачів у гнучкості, візуальності та адаптованості до різних сценаріїв використання. Незважаючи на широкий функціонал окремих аналогів, більшість з них мають обмеження, що стосуються візуалізації процесів, підтримки прикладного рівня протоколів, розширюваності та інтерактивності взаємодії з мережею.

Проведений аналіз архітектурних підходів дозволив обґрунтувати доцільність використання монолітної багаторівневої структури з розмежуванням логіки на окремі компоненти, що сприятиме структурованій реалізації проєкту на початковому етапі та створить основу для подальшої масштабованості програмного застосунку. Такий підхід забезпечує поєднання простоти реалізації, прозорості логіки та потенціалу до масштабування.

На основі аналітичних висновків були визначені завдання, які має охопити

проєкт. Йдеться про створення комплексного застосунку, що охоплює всі ключові аспекти симуляції мережевої взаємодії: побудову мережевої структури, обробку логіки передачі даних, конфігурацію параметрів та взаємодію користувача з моделлю.

Таким чином, розробка власного програмного симулятора є логічно вмотивованою і перспективною. Вона відкриває можливості для поєднання функціональних переваг відомих архітектур із реалізацією нового підходу до подання й вивчення роботи комп’ютерних мереж.

2 РОЗРОБКА МОДЕЛІ ТА АЛГОРИТМІВ ГРАФІЧНОГО СИМУЛЯТОРА

2.1 Аналіз вхідних даних симулятора

Аналіз вхідних даних для програмної реалізації симуляції мережевої взаємодії потребує визначення ключових інформаційних об'єктів, що формуються у процесі побудови топології, налаштування пристроїв та запуску симуляційних сценаріїв. Дані, з якими працює програмний застосунок, мають різноманітну, складну внутрішню структуру та високий ступінь взаємозалежності, що зумовлює необхідність системного підходу до їх організації, зберігання та обробки.

Для технічної реалізації симулятора обрано мову програмування Java [20], що у поєднанні з бібліотекою JavaFX [9] забезпечує побудову інтерактивного графічного інтерфейсу користувача та підтримку роботи з подієвою моделлю. Архітектурна організація симулятора ґрунтується на багаторівневій структурі з чітким розмежуванням відповідальності між логікою пристроїв, симуляційними процесами, конфігураційними параметрами та візуальним відображенням. Кожен функціональний компонент реалізується у межах власного модуля, що дозволяє розробляти, тестувати та модифікувати окремі частини застосунку незалежно. Такий підхід не лише спрощує підтримку коду та повторне використання логіки, а й забезпечує гнучкість при масштабуванні проєкту та інтеграції нових типів пристроїв і протоколів у майбутньому.

Виходячи з особливостей поставленого завдання, було визначено декілька основних модулів, кожен з яких відповідає за певний аспект функціонування симулятора: побудова топології, симуляція протоколів, конфігурація пристроїв, збереження проєкту, теоретична частина.

Модуль побудови мережевої топології працює з просторово-структурованими даними, що описують елементи мережі та зв’язки між ними. Основу цих даних становлять об’єкти пристроїв, які мають унікальні ідентифікатори, координати на робочому полі, типову класифікацію (ПК, маршрутизатор, комутатор, сервер) і список доступних інтерфейсів. Для кожного

інтерфейсу зберігається інформація про IP-адресу, MAC-адресу [14, 21], маску підмережі, шлюз та прив’язку до конкретного з’єднання. Дані про з’єднання, у свою чергу, визначають фізичну структуру мережі та фіксують, які інтерфейси підключені один до одного. Таким чином, уся топологія представляється як множина взаємозалежних об’єктів з просторовим позиціонуванням та логічними зв’язками, що формують основу симуляційного середовища.

Модуль симуляції оперує динамічними, даними механізму обробки подій, що змінюються в реальному часі відповідно до мережевої активності. До складу цих даних входять мережеві пакети різних типів, які містять службову та корисну інформацію: MAC- і IP-адреси відправника й одержувача, тип запиту, вкладений вміст тощо. Окрім самих пакетів, модуль обробляє події симуляції – передавання пакету між пристроями, очікування відповіді, відправлення оновлень протоколів тощо. Важливу роль відіграють таблиці стану, такі як ARP-кеші [22], таблиці маршрутизації, списки RIP-мереж [23], DNS-записи та інші проміжні структури, які зберігаються в пам’яті пристроїв і змінюються в процесі моделювання. Симуляція також вимагає контролю часу – для цього використовуються часові мітки, черги подій та механізми затримки доставки даних, що робить роботу модуля подібною до обробки повідомлень у реальній мережі.

Модулі конфігурації параметрів пристроїв працюють з параметрично структурованими даними, що задаються користувачем і визначають логіку роботи кожного окремого пристрою. До таких даних належать IP-адреси, маски підмереж, адреси шлюзів, режим призначення адрес, а також конфігураційні записи таблиць – наприклад, статичні маршрути з вказаною маскою та next-hop. Для серверів передбачена додаткова інформація: у DNS-серверів – це таблиця доменів і відповідних IP-адрес; у HTTP-серверів – набір доменів з відповідним HTML-вмістом [24], який передається клієнтам. Дані, що опрацьовуються цими модулями, є критично важливими для коректного функціонування протоколів і визначають результат симуляції. Зміни в цих параметрах призводять до перерахунку маршрутів, зміни логіки передавання, ініціації нових подій тощо.

Модуль збереження та завантаження проєктів відповідає за роботу зі зведеним набором даних, які відображають повний поточний стан симуляції. Серед них – топологія пристроїв, їхні позиції на робочому полі, всі параметри інтерфейсів, таблиці протоколів (ARP, маршрути, DNS), а також стан черги подій. Усі ці дані формують узгоджену модель, яку потрібно серіалізувати у формат, придатний для збереження – наприклад, JSON [25]. Завдяки цьому користувач має змогу зберігати, відкривати, копіювати та обмінюватися мережевими схемами, не втрачаючи жодного аспекту конфігурації. Цей модуль має працювати із забезпеченням цілісності структури даних, зокрема перевіряти відповідність між пристроями, інтерфейсами та з'єднаннями при імпорті.

Таким чином, аналіз вхідних даних симулятора комп'ютерної мережі вказує на необхідність впровадження п'яти функціональних модулів, кожен з яких оперує специфічними категоріями даних та потребує відповідного підходу до їх обробки і зберігання. Реалізація десктопного застосунку на базі Java з використанням JavaFX для інтерфейсної частини та об'єктно-орієнтованої моделі даних для логіки симуляції забезпечує міцну основу для побудови багаторівневої архітектури. Комплексна організація даних у вигляді взаємопов'язаних об'єктів із підтримкою серіалізації дозволяє досягти балансу між точністю симуляції, гнучкістю візуального представлення та можливістю подальшого масштабування. Завдяки такому підходу можна створити повноцінний інструмент, що сприятиме глибшому розумінню принципів мережевої взаємодії та протоколів у візуально доступній та інтерактивній формі.

2.2 Розробка моделі симулятора

Основу архітектури застосунку складає набір взаємопов'язаних модулів, реалізованих із використанням об'єктно-орієнтованого підходу. Уся логіка програми побудована на мові Java, що забезпечує повноцінну підтримку принципів ООП – наслідування, інкапсуляції, поліморфізму та абстракції [26]. Кожна частина функціональності представлена у вигляді окремого модуля, який

об’єднує певну групу класів, що спільно реалізують визначену область поведінки в межах симулятора.

Кожен модуль включає відповідні класи, які моделюють ключові елементи мережевого середовища. Наприклад, модуль побудови топології об’єднує класи, що описують пристрої, інтерфейси та з’єднання. Модуль симуляції складається з класів, які реалізують події, пакети та механізми обробки взаємодії між пристроями. Усі ці елементи організовані у вигляді класових ієрархій, де базові абстракції розширюються спеціалізованими підкласами для реалізації різних типів пристроїв, протоколів чи дій.

Такий підхід дозволяє досягти високого рівня модульності та гнучкості в архітектурі. Замість створення монолітної структури, проєкт розділено на логічні блоки, кожен з яких формує окремий модуль системи. Це, в свою чергу, спрощує супровід, полегшує тестування окремих компонентів, а також відкриває можливості для подальшого розширення – додавання нових типів пристроїв, протоколів або поведінки без зміни існуючої логіки інших частин системи.

Модуль, що відповідає за побудову топології мережі, формує логічну структуру взаємодії пристроїв. На цьому етапі користувач має змогу додавати до віртуального середовища об’єкти, які імітують комп’ютери, маршрутизатори, комутатори або сервери, розміщуючи їх у довільному порядку на віртуальному полі. Кожен пристрій може містити кілька інтерфейсів, які використовуються для встановлення з’єднань з іншими пристроями. Структура мережі формується шляхом створення зв’язків між інтерфейсами – фактично це фізичні або логічні канали передавання даних. Для кожного інтерфейсу вказуються мережеві параметри, включаючи фізичну адресу, IP-адресу, маску підмережі, шлюз та DNS. Завдяки цьому формується повноцінна віртуальна мережа, яка може функціонувати як модель для тестування взаємодії в локальній або навіть умовно глобальній мережі.

Модуль симуляції є ядром системи, що забезпечує опрацювання взаємодії пристроїв через мережу за допомогою стандартних мережевих протоколів. У межах цього модуля здійснюється генерація, маршрутизація та передача

мережевих пакетів між пристроями, відповідно до заданої топології та конфігурацій. Він оперує об’єктами, які моделюють пакети різного типу – запити до серверів, відповіді, контрольні пакети тощо. Кожен пакет має набір атрибутів, серед яких IP- і MAC-адреси відправника й одержувача, тип протоколу, інформаційне навантаження. Всі події в системі, пов’язані з передачею, формуються у вигляді окремих одиниць обробки, що організовані у чергу з урахуванням порядку виконання. Пристрої реагують на ці події, оновлюють власні таблиці стану (наприклад, ARP-кеші або таблиці маршрутів), формують відповіді та передають нові пакети далі. Такий механізм дозволяє імітувати як прості запити (наприклад, перевірку доступності), так і складніші взаємодії (запити до DNS- чи HTTP-серверів).

Модулі, що відповідають за конфігурацію пристроїв, надають користувачеві засоби для налаштування параметрів кожного пристрою в симульованій мережі. Через ці модулі задаються основні мережеві характеристики: IP-адреса, маска підмережі, адреса шлюзу, спосіб отримання IP, а також додаткові параметри, характерні для серверів – наприклад, таблиці доменів чи вміст відповіді на запити. Дані, які вводяться в рамках цього модуля, напряму впливають на логіку, за якою пристрої обробляють отримані пакети. Наприклад, відсутність правильного маршруту або запису в таблиці доменних імен може призвести до неможливості доставити пакет або до відмови в обробці запиту.

Модуль збереження та відновлення проєкту реалізує механізм серіалізації стану симуляції. Його завдання – зберігати всю необхідну інформацію, яка відображає повний поточний стан мережі: список створених пристроїв, їхні конфігурації, зв’язки, таблиці, що формуються в процесі роботи (наприклад, ARP або маршрутизаційні записи), а також активні події та положення об’єктів на полі. Збереження здійснюється у вигляді структури, яка зберігається у файл, а завантаження – шляхом відтворення всіх об’єктів і їхнього стану у початковому вигляді. Такий функціонал особливо важливий для створення навчальних

прикладів, сценаріїв демонстрацій або продовження роботи над раніше створеним проєктом.

У програмній системі симуляції мережевої взаємодії для збереження та відновлення конфігурацій використовуватиметься універсальний текстовий формат JSON, який є актуальним і загальноприйнятим стандартом обміну структурованими даними. Усі дані, що описують стан мережі – включаючи пристрої, інтерфейси, з'єднання, таблиці протоколів і події – зберігатимуться у цьому форматі. Для перетворення об'єктів Java у формат JSON і навпаки в системі використовується бібліотека Gson [27], яка дозволяє здійснювати серіалізацію та десеріалізацію без складної конфігурації. Завдяки цьому забезпечується можливість збереження повного стану симуляції у вигляді зрозумілого текстового файлу, що легко відновлюється для подальшої роботи або аналізу.

Теоретичний модуль забезпечує доступ до довідкової інформації, що супроводжує процес моделювання, та підтримує двомовний формат представлення теорії. Основу цього модуля становлять текстові дані, які завантажуються з ресурсних файлів і містять навчальний матеріал, та інтегрований інтелектуальний помічник завдання якого надання пояснень за запитами користувача в контексті мережевих технологій. Теоретичний матеріал представлено українською та англійською мовами паралельно, що дає змогу користувачу зіставляти технічну термінологію та сприяє формуванню профільної англомовної компетентності.

Розроблена модель симулятора забезпечує наочне представлення основних функціональних модулів та логічних взаємозв'язків між ними, що формує цілісне уявлення про архітектуру програмного забезпечення. Завдяки модульному підходу та об'єктно-орієнтованій реалізації, застосунок набуває структурованості, гнучкості та придатності до масштабування. Побудована модель слугує фундаментом для подальшої реалізації кожного компонента, визначаючи межі його відповідальності, спосіб взаємодії з іншими частинами

системи, а також забезпечуючи узгодженість і цілісність функціонування симулятора в цілому.

2.3 Розробка алгоритмів роботи симулятора

Функціонування симулятора комп'ютерної мережі базується на низці внутрішніх алгоритмів, які керують поведінкою пристроїв, маршрутизацією трафіку, передачею даних та візуалізацією процесів. Ці алгоритми визначають логіку взаємодії компонентів системи та реалізовують симуляцію на прикладному, мережевому, транспортному та канальному рівнях. Особливістю розробленого програмного застосунку є архітектура на основі подій, яка дозволяє запускати обробку мережевих подій залежно від дій користувача або внутрішнього перебігу симуляції.

Ключовими є алгоритми, пов'язані з імітацією клієнт-серверних запитів, обміну ARP-повідомленнями, маршрутизацією пакетів та обробкою прикладного HTTP-запиту. Вони тісно пов'язані між собою і моделюють типовий цикл мережевої взаємодії, що починається з дії користувача та завершується отриманням відповіді від сервера. Додатково, у симуляції реалізовано роботу протоколу автоматичної конфігурації пристроїв — DHCP, який дозволяє динамічно призначати IP-адреси клієнтам на основі запиту. Це значно спрощує початкову конфігурацію мережі, особливо у великих топологіях, і дозволяє моделювати типовий сценарій початкового з'єднання ПК із мережею.

Ще одним важливим компонентом є алгоритм динамічної маршрутизації на основі протоколу RIP (Routing Information Protocol) [23]. У рамках симуляції реалізовано підтримку обміну маршрутною інформацією між маршрутизаторами. Кожен маршрутизатор періодично надсилає іншим список відомих мереж і відстаней до них, що дозволяє автоматично оновлювати таблиці маршрутів у динамічних умовах.

Алгоритм роботи симуляції починається з того, що користувач через графічний інтерфейс створює мережу: додає пристрої (ПК, маршрутизатори, комутатори, сервери), з'єднує їх між собою, та виконує початкову конфігурацію

– задає IP-адреси, маски, шлюзи, тип маршрутизації. Кожен пристрій має набір інтерфейсів, і всі налаштування виконуються на рівні інтерфейсу, що дозволяє точно моделювати поведінку фізичних мережевих портів.

Після побудови топології користувач виконує певну дію, наприклад – вводить домен у вікні браузера на ПК. Ця дія ініціює процес, який охоплює одразу кілька алгоритмічних блоків. Спочатку система перевіряє, чи відома IP-адреса для введеного домену. Якщо ні – генерується DNS-запит. Цей запит створюється у вигляді симуляційного пакета, що потрапляє до черги подій. Перед його відправленням симулятор виконує маршрутизацію, тобто обирає, через який інтерфейс і в який бік відправити пакет, орієнтуючись на таблиці маршрутів.

Якщо мережа, до якої належить DNS-сервер, не є локальною, пристрій намагається надіслати пакет через шлюз. Тут спрацьовує алгоритм ARP-визначення – якщо MAC-адреса шлюзу не відома, створюється ARP-запит, який розсилається як broadcast у межах сегмента. Коли потрібний пристрій відповідає ARP-відповіддю, MAC-адреса кешується, і DNS-запит передається далі. При цьому всі пакети фізично «рухаються» мережею – вони послідовно потрапляють на інтерфейси пристроїв, які вирішують, що з ними робити: переслати далі, обробити або відкинути.

DNS-сервер, отримавши запит, звіряє його з власною таблицею доменних записів, формує відповідь і повертає її назад – маршрутом, що обертається автоматично відповідно до IP-полів пакета. Коли ПК отримує IP-адресу, він зберігає її і переходить до наступного етапу – формування HTTP-запиту. Цей запит відправляється аналогічно: спочатку перевіряється маршрут, при потребі оновлюються ARP-записи, пакет передається через мережу до сервера. Сервер, отримавши запит, відправляє відповідь із вмістом сторінки.

Важливим компонентом у всьому процесі є черга подій симуляції. Кожна подія – це об’єкт, який описує дію у мережі: надсилання пакета, очікування відповіді, оновлення таблиці, візуалізацію на полі. Події можуть бути синхронними або відкладеними. Усі пристрої симулятора реагують лише на ті

події, які їх стосуються, та змінюють власний внутрішній стан – кеші, таблиці, черги пакетів, а також виводять інформацію у лог чи інтерфейс.

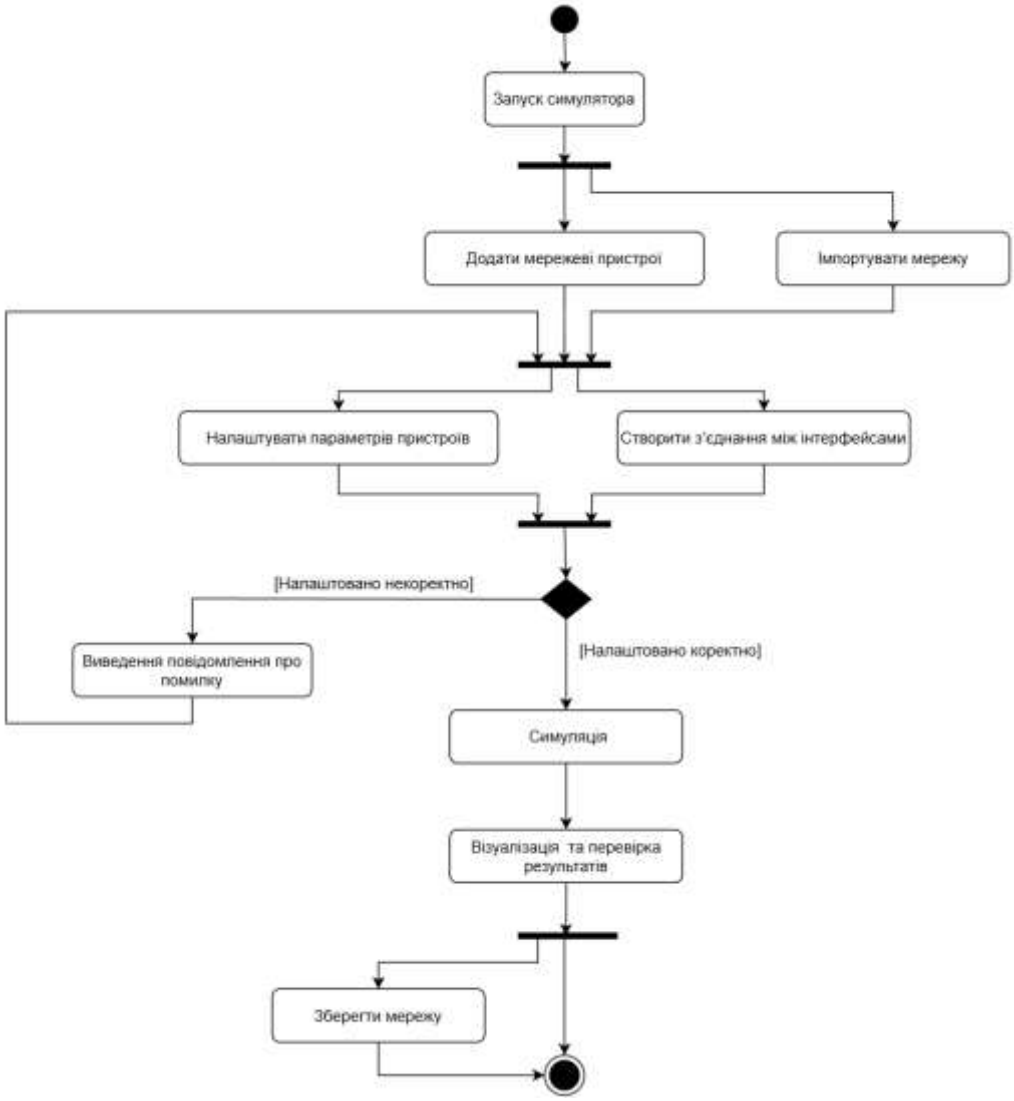


Рисунок 2.1 – Діаграма діяльності

Діаграма діяльності, розміщена на рисунку 2.1, відображає узагальнений алгоритм взаємодії користувача з симулятором. Вона починається з запуску програми, включає побудову мережі, налаштування пристроїв, запуск симуляції, генерацію подій, обробку запитів і виведення результатів. Блок-схема (див. рисунок 2.2) деталізує етапи передачі HTTP-запиту через мережу з урахуванням попереднього DNS-визначення. Вона демонструє залежності між ARP, маршрутизацією, симуляцією подій та прикладною логікою.

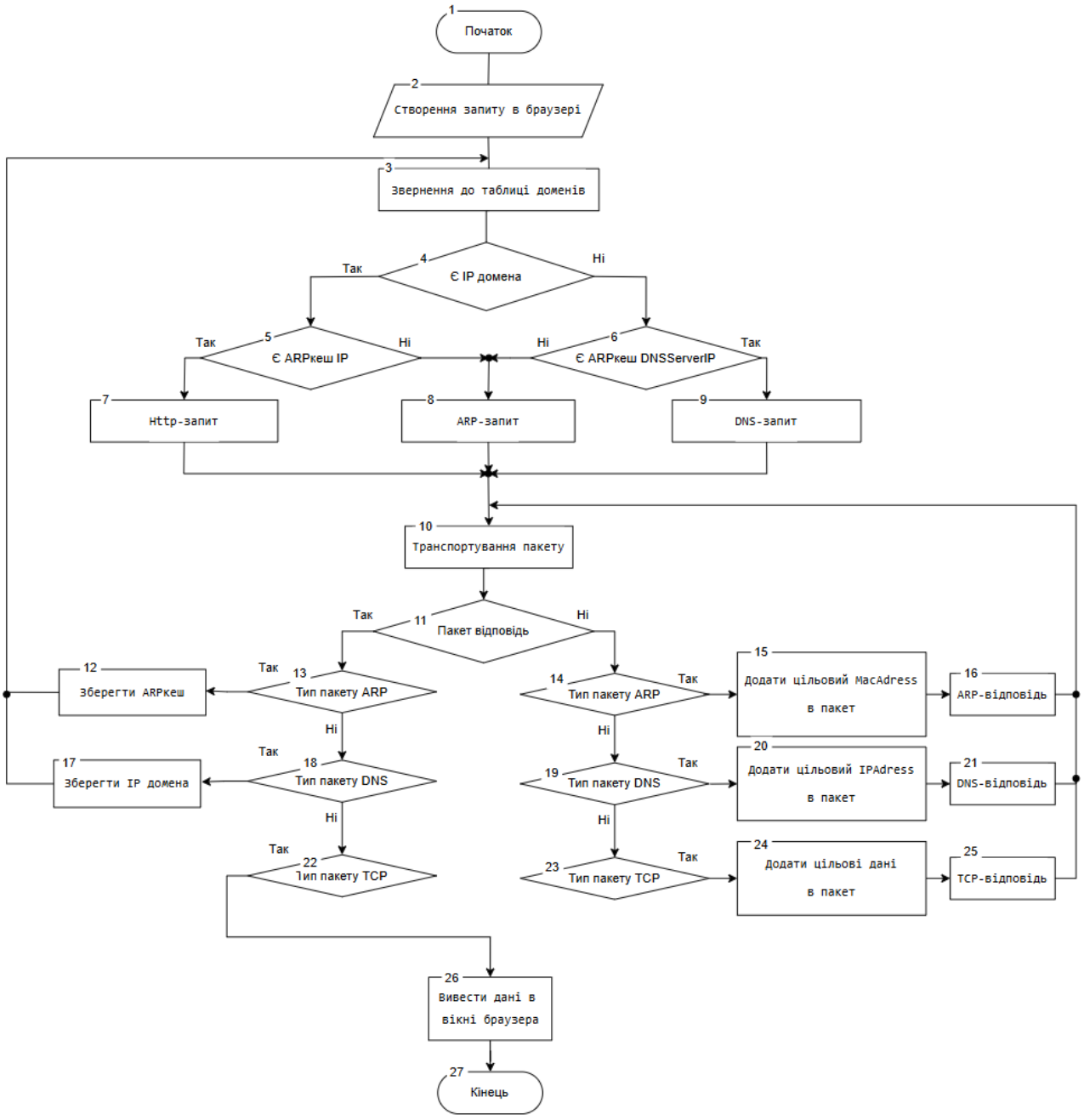


Рисунок 2.2 – Блок-схема передачі HTTP-запиту

Отже, розроблені алгоритми забезпечують повноцінне моделювання процесів передачі даних у комп’ютерній мережі. Вони демонструють тісну взаємодію між логікою протоколів, конфігурацією пристроїв, поведінкою симуляції та інтерфейсом користувача. Завдяки використанню механізу обробки подій, реалізація є гнучкою, масштабованою та дозволяє наочно вивчати динаміку роботи мережевих служб.

2.4 Висновки

У результаті проведеного аналізу було сформовано архітектурну модель симулятора комп’ютерної мережі, яка базується на об’єктно-орієнтованому підході та модульному принципі побудови. Система складається з чотирьох функціональних модулів, кожен з яких оперує власним набором даних і реалізує чітко визначену частину логіки: побудову мережевої топології, обробку симуляційних подій, конфігурацію параметрів пристроїв, збереження та завантаження проєктів, а також візуалізацію взаємодії користувача з симульованим середовищем.

У центрі реалізації лежить механізм симуляції подій, що забезпечує поступову обробку мережевих запитів та реакцій пристроїв відповідно до правил роботи реальних протоколів. Було розроблено й детально описано алгоритми, які визначають логіку виконання типових сценаріїв мережевої взаємодії. Ці алгоритми охоплюють усі ключові аспекти роботи мережі: від фізичної адресації до логіки клієнт-серверної взаємодії.

Таким чином, розроблена система забезпечує не лише базову функціональність симуляції, а й демонструє гнучкість та адаптивність до складніших завдань. Запропонована архітектура є надійною основою для подальшої розробки проєкту в напрямках розширення мережевих протоколів, додавання механізмів тестування, автоматизації або інструментів навчального аналізу мережевої поведінки.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ РЕАЛІЗАЦІЇ ГРАФІЧНОГО СИМУЛЯТОРА ІНТЕРНЕТ МЕРЕЖІ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного забезпечення

Розробка системи симуляції комп'ютерної мережі вимагає відповідального підходу до вибору мов програмування, бібліотек, фреймворків та інструментів, які забезпечать стабільність, розширюваність і гнучкість програмного забезпечення. З урахуванням специфіки функціонування симулятора, зокрема моделювання обміну мережевими пакетами, підтримки протоколів, обробки подій та візуалізації мережевих процесів, були обрані технології, що повністю відповідають цим вимогам.

Мовою програмування для реалізації симулятора було обрано Java, яка надає широкі можливості для створення об'єктно-орієнтованих систем з високим рівнем надійності. Саме об'єктна модель Java дозволяє реалізувати кожен компонент симуляції – пристрої, інтерфейси, пакети, події – у вигляді класів зі спадковою ієрархією. Це дає змогу чітко розмежовувати призначення між частинами системи, забезпечувати відсутність повторного використання коду та спрощувати його підтримку.

Для побудови графічного інтерфейсу обрано JavaFX, який підтримує гнучку інтеграцію із внутрішньою логікою симулятора та дозволяє створити візуальне представлення топології мережі, анімацію переміщення пакетів та динамічне оновлення стану пристроїв. JavaFX забезпечує високу інтерактивність середовища та зручність користувача під час роботи з пристроями, з'єднаннями, параметрами конфігурації та переглядом результатів симуляції. Завдяки цьому інструменту реалізовано модуль візуального інтерфейсу, що є основним каналом взаємодії користувача з системою.

У застосунку застосовується архітектура з обробкою подій керування процесами, що реалізована через механізм черги подій. Усі симуляційні сценарії – від ARP-запитів до HTTP-взаємодії – формуються у вигляді подій, які

обробляються у визначеному порядку. Такий підхід забезпечує реалізм у відображенні часових затримок, маршрутизації, відповіді на запити та реакцій мережевих пристроїв.

Для збереження конфігурації мережі, включаючи розташування пристроїв, параметри інтерфейсів, стан таблиць і чергу подій, використовується формат JSON. Усі дані серіалізуються та десеріалізуються за допомогою бібліотеки Gson, яка забезпечує просте перетворення об’єктів Java у формат, зручний для збереження, передачі та відновлення. Це дозволяє користувачеві зберігати створені топології, повторно їх завантажувати та працювати над симуляціями в різних сеансах.

Загалом, поєднання Java, JavaFX, об’єктно-орієнтованої моделі програмування, серіалізації у форматі JSON та модульної архітектури дозволило створити гнучку, розширювану та ефективну платформу для моделювання комп’ютерних мереж. Обрані технології забезпечують як технічну надійність реалізації, так і зручність для кінцевого користувача у процесі побудови мережі, конфігурування пристроїв і аналізу результатів симуляції.

3.2 Розробка модуля симуляції

Модуль симуляції відповідає за обробку динамічних мережевих подій та моделювання передачі даних у віртуальному середовищі. Він реалізує логіку взаємодії між пристроями відповідно до принципів мережевої моделі OSI, забезпечуючи підтримку типових протоколів різних рівнів – від канального до прикладного. Центральним елементом цього модуля є механізм управління чергою подій, у якій зберігаються всі активні дії, що відбуваються в системі під час симуляції.

Кожна дія в системі представлена як подія – наприклад, надсилання або приймання пакета, запуск таймера, оновлення таблиці маршрутизації або ARP. Ці події додаються до черги та обробляються у порядку надходження або відповідно до заданої затримки. Події можуть породжувати нові події,

утворюючи логічний ланцюг взаємодії, який точно відтворює поведінку мережі в часі.

На рисунку 3.1 зображено діаграму асоціації подій та їх черги модуля симуляції.

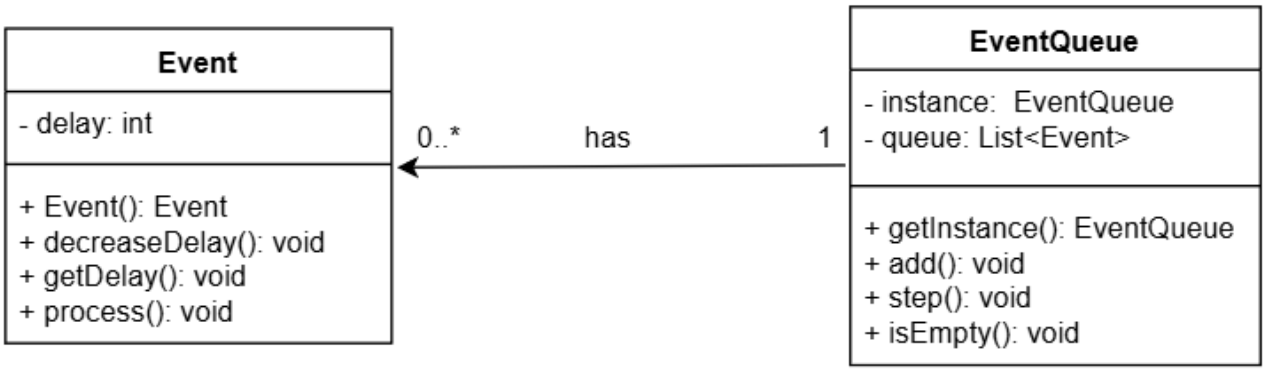


Рисунок 3.1 – Діаграма асоціації подій

Модуль підтримує реалізацію низки протоколів:

– Address Resolution Protocol для визначення MAC-адрес;

– ICMP (Ping) для перевірки доступності вузлів [28];

– Routing Information Protocol для оновлення таблиць маршрутів;

– Domain Name System для резолюції доменних імен;

– HTTP для моделювання прикладного трафіку та взаємодії клієнта з веб-сервером.

Кожен пакет у симуляції містить структуровані поля: адреси джерела та одержувача, тип пакета, вміст (у разі прикладного запиту) та службову інформацію. Пакет передається від пристрою до пристрою через мережу, при цьому система враховує таблиці маршрутизації, ARP-кеші, типи з’єднань та конфігурацію інтерфейсів. У разі відсутності потрібних записів автоматично генеруються допоміжні запити, наприклад ARP або DNS.

На рисунку 3.2 зображено діаграму класів модуля симуляції, зокрема підмодуля packet, який відповідає за моделювання мережевих пакетів.

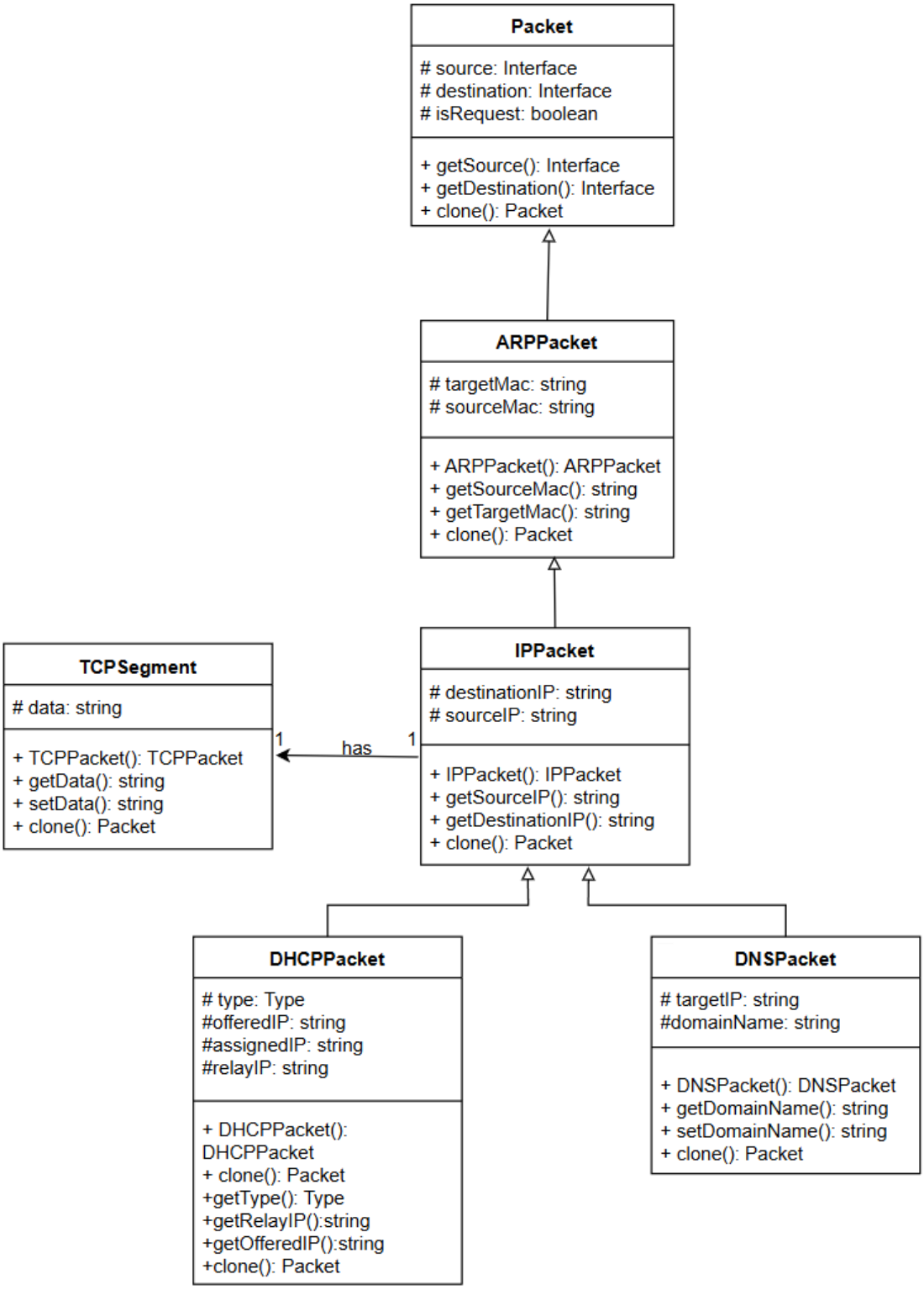


Рисунок 3.2 – Діаграма класів пакетів

Отже, клас-нащадок останнього рівня DHCPPacket, що спеціалізується на передачі службових повідомлень для динамічного отримання IP-адреси в межах симульованого мережевого середовища, містить такі параметри:

– source – інтерфейс або пристрій, що ініціював надсилання DHCP-пакета;

– destination – інтерфейс або вузол, який є отримувачем DHCP-пакета;

– isRequest – булевий параметр, що вказує на тип пакета;

– targetMac – MAC-адреса призначення, до якої спрямовується пакет на канальному рівні;

– sourceMac – MAC-адреса джерела, з якої було надіслано пакет;

– sourceIP – IP-адреса відправника пакета;

– destinationIP – IP-адреса призначення;

– type – тип DHCP-пакета;

– offeredIP – IP-адреса, яку DHCP-сервер пропонує клієнту для використання;

– assignedIP – IP-адреса, фактично призначена клієнту;

– relayIP – IP-адреса DHCP Relay Agent'a, якщо DHCP-пакет передається через проміжний маршрутизатор.

А також методи отримання та присвоєння значень параметрів (геттери та сеттери), клонування й конструктор.

Алгоритм обробки подій повністю відокремлений від графічного інтерфейсу, що дозволяє запускати симуляцію як у покроковому, так і в автоматичному режимі. Користувач може спостерігати за виконанням подій у реальному часі або аналізувати результат після завершення симуляції. Така ізоляція забезпечує гнучкість архітектури та полегшує тестування.

3.3 Розробка модуля побудови мережевої топології

Модуль побудови мережевої топології є початковим етапом у роботі з симулятором та забезпечує створення логічної моделі комп’ютерної мережі. Основним завданням модуля є додавання мережевих пристроїв, організація з’єднань між ними та формування повноцінної структури для подальшого моделювання процесів передачі даних. Побудова топології відбувається у

візуальному середовищі, де кожен пристрій представлений як об’єкт з чіткими координатами на полі та внутрішнім набором параметрів.

Можна додати на поле наступні типи пристроїв: персональні комп’ютери, маршрутизатори, комутатори, а також сервери з різною функціональністю – DNS-сервери та HTTP-сервери. Кожен пристрій створюється як самостійна сутність і має унікальний ідентифікатор, що дозволяє відслідковувати його в межах системи. Усі пристрої автоматично додаються до загального списку активної топології, а також прив’язуються до графічного представлення.

Особливістю реалізації є те, що кожен пристрій містить один або кілька інтерфейсів, через які здійснюється підключення до інших пристроїв. Інтерфейси мають власні MAC-адреси, IP-адреси, маски підмереж, шлюзи та інші мережеві параметри. Структура модуля дозволяє підключати інтерфейси між собою, створюючи логічні лінії зв’язку, які моделюють фізичні кабелі. У процесі побудови мережі користувач може обрати конкретні інтерфейси для підключення, тим самим чітко визначаючи, які порти використовуються у симуляції.

З’єднання між інтерфейсами створюються вручну та мають двосторонній характер. Симулятор автоматично слідкує за тим, щоб один інтерфейс не був підключений до кількох інших одночасно, що відповідає правилам реальної мережевої інфраструктури. Якщо користувач підключає вже задіяний інтерфейс до нового з’єднання, попереднє автоматично видаляється. Крім того, передбачена можливість від’єднання інтерфейсів шляхом видалення з’єднання вручну через взаємодію з візуальним інтерфейсом.

На рівні внутрішніх структур модуль працює з об’єктами, які зберігають повну інформацію про тип пристрою, його ім’я, координати на полі, список інтерфейсів і їх зв’язки. Усі ці дані взаємопов’язані та підтримують узгодженість топології. Зміна позиції пристрою на полі автоматично оновлює координати з’єднань, що забезпечує коректне відображення всієї схеми в графічному середовищі.

Модуль побудови топології тісно інтегрований з модулями конфігурації та симуляції. Після завершення побудови користувач має можливість перейти до налаштування кожного пристрою через відповідне вікно, де вводяться IP-параметри, вибирається режим маршрутизації або визначаються таблиці DNS/HTTP для серверів. Побудована схема використовується модулем симуляції як вхідні дані для розрахунку маршрутів, формування запитів, передавання пакетів та обробки подій.

3.4 Розробка модуля конфігурації пристроїв

Модуль конфігурації пристроїв відповідає за зберігання і обробку параметрів мережевих елементів, що є частиною побудованої топології. Його основна функція полягає у забезпеченні гнучкого визначення мережевої поведінки кожного пристрою через доступ до конфігураційних структур, що зберігають ключові параметри для подальшої симуляції.

Кожен пристрій у симуляторі має одну або кілька конфігурованих мережевих інтерфейсів. Для кожного інтерфейсу задаються основні мережеві атрибути, включаючи IP-адресу, маску підмережі, адресу шлюзу, DNS-сервер, а також унікальну MAC-адресу. Параметри зберігаються у спеціальних об’єктних структурах і безпосередньо використовуються при моделюванні таких процесів як ARP-визначення, маршрутизація, DNS-запити та обробка HTTP-трафіку. Це дозволяє симулятору відтворювати логіку роботи IP-мереж на рівні симуляції.

Для маршрутизаторів реалізовано зберігання типу маршрутизації та відповідних маршрутних записів. У разі статичної маршрутизації пристрій зберігає список вручну доданих маршрутів із зазначенням підмережі, маски та наступного хопа. У динамічному режимі, який імітує роботу протоколу RIP, зберігаються анонсовані мережі, які пристрій поширює до сусідів у симуляції.

Для серверів, таких як DNS та HTTP, також передбачено структури для конфігурації служб. У DNS-серверів зберігається таблиця записів, яка встановлює відповідність між доменними іменами та IP-адресами, і використовується під час обробки DNS-запитів. HTTP-сервери містять таблиці

віртуальних доменів, до кожного з яких прив’язаний симульований вміст сторінки. Ці дані є основою для моделювання відповіді на HTTP-запити, що надходять від клієнтів.

На рисунку 3.3 подано діагрму класів, які відповідають за процес конфігурації мережевих пристроїв.

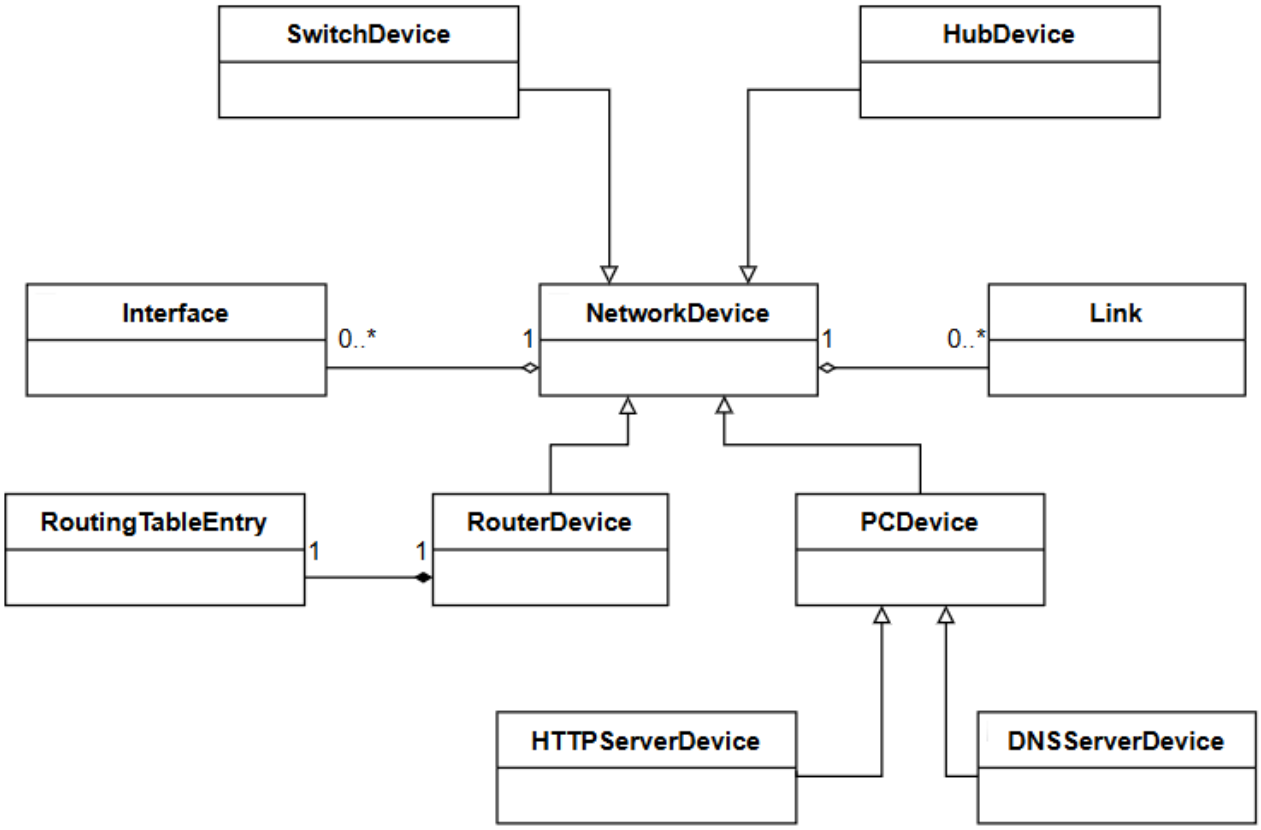


Рисунок 3.3 – Діаграма класів модуля конфігурації пристроїв

Кожен пристрій також має унікальну назву, яка задається при створенні та використовується для логування, виводу в інтерфейсі та підписування у візуальному представленні мережі. Назва зберігається окремо і не впливає на технічні параметри, однак є важливою для навігації в складних схемах.

3.5 Розробка модуля збереження та завантаження проєктів

Модуль збереження та завантаження проєктів відповідає за фіксацію поточного стану мережі у вигляді структурованого об’єкта та подальше його

відновлення у вигляді повноцінної мережевої топології. Цей модуль забезпечує збереження всієї симуляційної моделі, включно з пристроями, інтерфейсами, з’єднаннями, мережевими конфігураціями та всіма супутніми даними, які були створені або змінені під час побудови та налаштування мережі.

Збереження відбувається шляхом формування спеціальної структури, що відображає всю логіку створеної моделі. Вона включає:

– список пристроїв з унікальними ідентифікаторами;

– позиції пристроїв на графічному полі;

– набір інтерфейсів з усіма параметрами;

– перелік активних з’єднань між інтерфейсами;

– конфігураційні параметри для кожного пристрою, включно з таблицями маршрутів, DNS-записами, HTTP-вмістом.

Ця структура серіалізується у формат JSON за допомогою бібліотеки Gson, що дозволяє зберегти дані у зручному, текстовому, переносному вигляді. JSON-файл з проєктом можна зберігати локально, переносити між системами, зберігати в архіві або ділитись з іншими користувачами.

Під час завантаження проєкту застосунок виконує десеріалізацію JSON-структури та автоматично відтворює всю топологію. Всі пристрої створюються повторно, розміщуються на полі згідно з координатами, відновлюються з’єднання між інтерфейсами, повертаються початкові IP-конфігурації, а також відновлюються всі допоміжні таблиці – маршрути, ARP, DNS, HTTP. Таким чином, користувач отримує повну копію попередньої симуляції з можливістю одразу її запуску або редагування.

Окремо варто зазначити, що при відновленні проєкту всі логічні зв’язки між пристроями та інтерфейсами зберігаються. Завдяки цьому після завантаження не потрібно повторно встановлювати з’єднання або перенастроювати мережу. Це дозволяє використовувати модуль як інструмент для підготовки навчальних схем, тестування сценаріїв або створення шаблонів для симуляції.

На рисунку 3.4 зображено діаграму класів модуля збереження та завантаження проєктів.

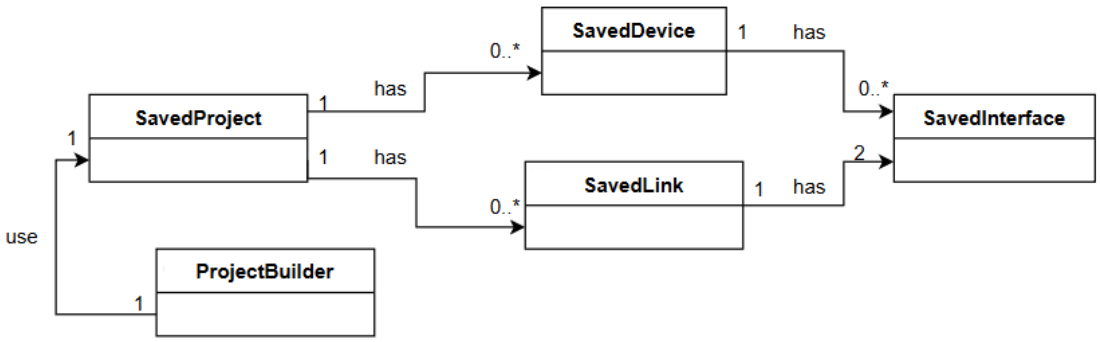


Рисунок 3.4 – Діаграма асоціації класів модуля збереження та завантаження проєктів

Модуль інтегрується з іншими компонентами системи: дані, які вводяться через модуль конфігурації, зберігаються при експорті проєкту, а після імпорту автоматично враховуються під час виконання алгоритмів симуляції. Це забезпечує безперервність роботи симулятора навіть при багаторазовому завантаженні та збереженні.

3.6 Розробка модуля отримання та відображення теоретичної інформації

У межах реалізації теоретичного модуля графічного симулятора комп'ютерної мережі була створена функціональність, що дозволяє користувачу звертатися до мовної моделі ChatGPT [29]для отримання пояснень, уточнень або розширеного опису теоретичних понять. Такий інструмент інтегровано безпосередньо в інтерфейс симулятора, де користувач може у будь-який момент поставити запитання та миттєво отримати відповідь на нього у вигляді текстового повідомлення.

Робота цього модуля базується на взаємодії з офіційним API [30], який надає компанія OpenAI [29]. Для формування та надсилання HTTP-запитів у програмі використовується стандартна бібліотека HTTPClient [31]. Вона

забезпечує зручну та безпечну взаємодію з зовнішнім сервером OpenAI. Сам запит містить текстове повідомлення користувача та параметри виклику мовної моделі, включаючи вибір конкретної моделі.

Формат передавання даних між клієнтом і сервером побудований на основі JSON. Для серіалізації та десеріалізації цих даних у програмі застосовується бібліотека Jackson, а саме її компонент ObjectMapper, що дозволяє перетворювати Java-об’єкти у JSON-рядки та навпаки. Це спрощує роботу з тілом запиту і відповіді, дозволяючи ефективно працювати з вкладеними структурами даних.

Після формування запиту він надсилається на сервер OpenAI, де модель обробляє введений текст і формує відповідь. Отримана відповідь повертається у вигляді JSON-структури, з якої витягується згенерований текст і виводиться на екран у вигляді повідомлення. Візуально це виглядає як чат-інтерфейс.

У реалізації модуля передбачено обробку типових помилок, які можуть виникнути під час запиту: проблеми з мережею, неправильний API-ключ, перевищення лімітів запитів тощо. У таких випадках користувач отримує зрозуміле повідомлення з поясненням причини, що дозволяє уникнути критичних збоїв у роботі програми. Це робить систему більш стабільною та зручною в експлуатації.

У графічному інтерфейсі застосунку модуль представлено у вигляді окремої області, де користувач може вводити свої запити, переглядати історію діалогу та перемикатися між мовами. Відповіді надаються українською або англійською мовами залежно від налаштувань або змісту запиту. Таким чином, модуль також виконує функцію перекладача термінів і сприяє формуванню технічної англомовної компетентності.

У межах реалізації теоретичного модуля графічного симулятора також було створено окреме вікно, у якому розміщено два залежні текстові поля. Перше з них містить україномовну версію теоретичного матеріалу, а друге – англомовну. Обидва тексти завантажуються з внутрішніх ресурсів застосунку, що забезпечує автономність роботи та не потребує підключення до мережі.

Текстові області синхронізовані за вертикальною прокруткою. Це дозволяє користувачу зіставляти потрібні фрагменти тексту двома мовами, для легшого засвоєння користувачем матеріалу на іноземній мові. Крім того реалізовано пошукове поле, що дозволяє виконати текстовий пошук одночасно в обох мовних версіях. Введене користувачем ключове слово автоматично підсвічується в обох текстових полях, що полегшує навігацію в документі та дозволяє швидко знаходити потрібну інформацію в обох мовах.

3.7 Розробка графічного інтерфейсу програмного застосунку

Графічний інтерфейс симулятора мережі реалізовано з використанням бібліотеки JavaFX, що дозволяє створити інтерактивне середовище для візуалізації топології, відображення пристроїв, з’єднань та симуляційних подій. Основною метою інтерфейсу є забезпечення зручного візуального контролю над структурою мережі, параметрами пристроїв і перебігом процесів передачі даних.

Візуальне поле, що займає центральну частину вікна застосунку, є областю для побудови мережевої схеми. На цьому полі користувач розміщує мережеві пристрої, які представлені у вигляді інтерактивних елементів з унікальними підписами. Кожен пристрій можна перетягувати, редагувати його положення та встановлювати з’єднання з іншими елементами через вказані інтерфейси. Візуальні лінії з’єднань автоматично оновлюються при зміні координат, зберігаючи цілісність топології.

Крім розміщення пристроїв, інтерфейс відображає візуальне проходження мережевих пакетів. Під час симуляції на екрані можна спостерігати, як пакети «рухаються» від одного пристрою до іншого, що дозволяє користувачеві краще розуміти логіку маршрутизації, ARP-визначення або передачу DNS- та HTTP-запитів. Усі події анімовано з урахуванням напрямку та типу трафіку.

Основні елементи керування розташовано у верхній частині вікна застосунку. Тут доступні кнопки для додавання пристроїв, запуску або зупинки симуляції, збереження та завантаження проєктів. Кожна дія виконується через

відповідні інтерактивні кнопки, що забезпечує просту та зрозумілу взаємодію з програмою.

Графічна схема головного вікна зображена на рисунку 3.5.

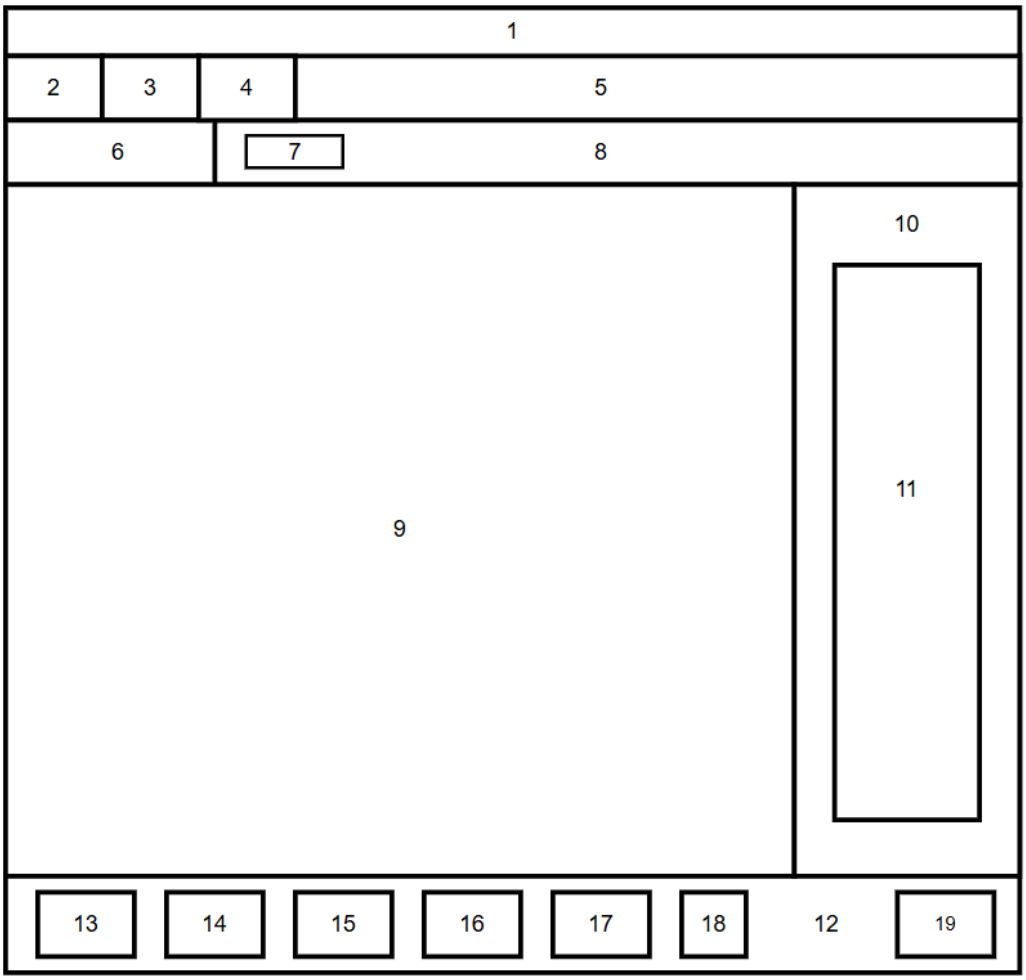


Рисунок 3.5 – Графічна схема головного вікна

Елементи головного вікна:

1. primaryStage – основне вікно застосунку з назвою, іконкою та всіма елементами інтерфейсу.

2. fileMenu – меню «Файл».

3. referenceMenu – меню «Довідка».

4. infoMenu – меню «Інфо».

5. menuBar – головна панель меню.

6. speedSlider – повзунок для регулювання швидкості симуляції.

7. connectBtn – кнопка «З’єднати» для створення з’єднання між пристроями.

8. topPanel – вертикальна панель, яка містить menuBar і controlPanel.

9. simulationPane – основне поле моделювання, де розміщуються пристрої.

10. logPanel – вертикальна панель, що містить logArea з заголовком «Логи».

11. logArea – текстова область для журналу подій (логів).

12. elementIconsBox – контейнер із кнопками створення пристроїв.

13. elementsLabel – напис «Доступні елементи».

14. vboxpc – кнопка для створення комп’ютера.

15. vboxrouter – кнопка для створення роутера.

16. vboxswitch – кнопка для створення свіча.

17. vboxhub – кнопка для створення хаба.

18. vboxserver – кнопка для створення сервера.

19. timerLabel – мітка, яка показує час симуляції.

У правій частині інтерфейсу розміщено лог подій. У ньому виводяться текстові повідомлення про всі ключові етапи симуляції: створення та обробку пакетів, відповіді пристроїв, ARP-запити, отримані DNS-відповіді тощо. Лог подій дозволяє користувачеві проаналізувати хід симуляції та швидко виявити можливі помилки у конфігурації мережі.

У кожного пристрою, доданого на поле симуляції, передбачено окреме вікно налаштувань, яке відкривається при взаємодії з елементом. Це вікно забезпечує доступ до ключових параметрів, які впливають на поведінку пристрою під час симуляції. Конфігураційне вікно реалізовано у вигляді інтерфейсу з вкладками, де кожна вкладка відповідає за певну категорію налаштувань.

Базовою є вкладка «Мережа», яка містить поля для налаштування IP-адреси, маски підмережі, шлюзу за замовчуванням та DNS-сервера. Також тут відображається MAC-адреса інтерфейсу, яка генерується автоматично та не підлягає редагуванню. Користувач має змогу обирати режим IP-конфігурації –

статичний або динамічний. Усі ці параметри задаються окремо для кожного інтерфейсу пристрою, що дозволяє моделювати складніші сценарії з декількома мережевими підключеннями.

У вкладці «Назва пристрою» можна змінити ім’я пристрою, яке відображається безпосередньо на полі моделювання та в журналі подій. Це покращує зручність у роботі з великими топологіями, дозволяючи швидко ідентифікувати вузли.

Вікно також містить вкладку «Термінал», яка виконує роль симуляційного виводу. Тут відображається текстова інформація, пов’язана з конфігурацією пристрою, подіями або результатами тестових дій. Хоча на даному етапі реалізація вкладки базується лише на виводі, вона відіграє допоміжну роль у візуалізації внутрішніх процесів.

Для серверів реалізовані додаткові вкладки, які відображають специфічні функції служб. У DNS-сервера присутня вкладка «DNS таблиця», де задається відповідність доменних імен до IP-адрес. Ці дані використовуються під час симуляції DNS-запитів. HTTP-сервер має вкладку «HTTP вміст», яка дозволяє задавати текст HTML-відповіді для різних віртуальних доменів – саме ці відповіді надсилаються клієнтам у відповідь на HTTP-запити.

Маршрутизатори, крім загальних мережевих параметрів, мають окрему вкладку для налаштування типу маршрутизації. Користувач може обрати між статичною маршрутизацією, де маршрути додаються вручну, та динамічною (RIP), у якій задаються мережі, що мають бути анонсовані іншим маршрутизаторам. Усі дані зберігаються у таблицях маршрутизації пристрою та використовуються модулем симуляції для направлення пакетів. Графічна система побудована на принципах модульності. Кожен тип пристрою має власний візуальний представник та стиль відображення. Це дозволяє швидко орієнтуватися в структурі мережі та спрощує її логічний аналіз.

3.8 Висновки

Отже, було обґрунтовано вибір технологій та інструментів, що використовувалися для розробки модулів симулятора комп’ютерної мережі. Для реалізації програмного забезпечення обрано мову програмування Java, яка забезпечує повноцінну підтримку об’єктно-орієнтованого підходу та дозволяє моделювати мережеві об’єкти у вигляді взаємопов’язаних класів. Для створення графічного інтерфейсу було використано бібліотеку JavaFX, що дозволила реалізувати динамічне візуальне середовище з інтерактивними елементами, які відображають пристрої, з’єднання та процеси передачі даних. У розділі детально описано чотири основні модулі системи: модуль побудови топології, модуль симуляції, модуль конфігурації пристроїв, модуль збереження та завантаження проєктів, а також графічний інтерфейс. Для кожного з них розглянуто архітектурні принципи побудови, структуру даних і взаємозв’язки з іншими компонентами симулятора. Інтеграція вікон налаштування для кожного пристрою дозволила забезпечити повний цикл взаємодії – від побудови до запуску мережевого сценарію, з підтримкою конфігурації IP, таблиць маршрутів, доменних записів та HTTP-відповідей. Всі рішення реалізовані у рамках єдиної архітектури, що забезпечує модульність, розширюваність і зручність у використанні.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАСТОСУНКУ

4.1 Тестування графічного симулятора

Тестування програмного забезпечення є важливим етапом життєвого циклу розробки, який має на меті виявлення невідповідностей між очікуваною та фактичною поведінкою системи. В межах цього проєкту тестування проводилось із фокусом на ручні сценарії, що дозволяють дослідити функціональність симулятора з точки зору кінцевого користувача, перевірити коректність роботи мережевих протоколів, адекватність графічного інтерфейсу та точність симуляційних алгоритмів.

Уся перевірка здійснювалася у режимі ручного тестування без використання автоматизованих фреймворків. Це дало можливість перевірити стабільність графічної частини, логіку взаємодії пристроїв, відповідність конфігурацій фактичній поведінці, а також виявити типові помилки у відображенні подій симуляції. Враховуючи симуляційну специфіку системи, ручне тестування виявилось доцільним методом перевірки.

Першим етапом тестування стала перевірка додавання пристроїв та з’єднань. На візуальне поле було додано різні типи пристроїв (ПК, маршрутизатор, комутатор, сервери), після чого перевірялась можливість їх переміщення, створення з’єднань між інтерфейсами, а також автоматичне оновлення ліній. Встановлено, що всі пристрої мають унікальні ідентифікатори, інтерфейси не допускають дублювання з’єднань, а візуальні лінії правильно оновлюються при зміні положення пристроїв.

Другим етапом стала перевірка вікон налаштувань пристроїв. Було відкрито вікна для ПК, маршрутизатора, DNS- і HTTP-серверів. Тестувалася можливість зміни IP-адрес, маски, шлюзу, DNS-сервера та назви пристрою. Для серверів перевірялась робота вкладок DNS-записів та HTTP-вмісту. Усі дані після зміни зберігались у структурі пристрою та використовувались модулем симуляції, що підтвердило коректну взаємодію між інтерфейсом та логікою.

Наступним етапом стала перевірка роботи основних протоколів. Створено топології, у яких реалізовано передачу ICMP (Ping), запити DNS, HTTP та обмін ARP. У випадках, коли MAC-адреса не була відома, симулятор автоматично створював ARP-запит і правильно формував ARP-відповідь. При введенні доменного імені у вікні браузера генерувався DNS-запит, а при його отриманні – створювався HTTP-запит до відповідного сервера. Усі ці процеси відображались як події у черзі симуляції, і супроводжувались візуалізацією передачі пакетів на полі.

Було перевірено обробку таблиць маршрутизації. У конфігураціях маршрутизаторів задавались статичні маршрути. Змодельовано сценарій, де шлях до сервера був можливий лише через кілька проміжних вузлів, і було підтверджено, що пакети успішно маршрутизувались згідно з таблицями.

Наступним етапом стала перевірка роботи основних протоколів. Створено топології, у яких реалізовано передачу ICMP (Ping), запити DNS, HTTP та обмін ARP. У випадках, коли MAC-адреса не була відома, система автоматично створювала ARP-запит і правильно формувала ARP-відповідь. При введенні доменного імені у вікні браузера генерувався DNS-запит, а при його отриманні – створювався HTTP-запит до відповідного сервера. Окрім цього, у рамках тестування перевірено роботу протоколу DHCP: пристрої без попередньо заданих IP-адрес успішно отримували конфігурацію від DHCP-сервера. Також перевірено обмін маршрутною інформацією між маршрутизаторами за допомогою протоколу RIP – симулятор коректно формував оновлення маршрутів та забезпечував динамічну побудову таблиць маршрутизації. Усі ці процеси відображались як події у черзі симуляції та супроводжувались візуалізацією передачі пакетів на полі.

Окрему увагу приділено функціональності збереження та завантаження проєкту. Після побудови повної топології мережі, включно з конфігураціями та з’єднаннями, проєкт було збережено у форматі JSON. Потім виконувалося завантаження з цього файлу. Тестування показало, що всі елементи мережі, їхні параметри, з’єднання та позиції було відновлено коректно, що підтверджує

надійність модуля збереження. Також було проведено суб’єктивне тестування продуктивності. Під час перевірки створювались великі топології з десятків пристроїв та сотень подій. Визначено, що симулятор підтримує комфортний час відгуку, не сповільнюється при симуляції складних сценаріїв і не допускає візуальних затримок у відображенні передачі пакетів або зміни станів пристроїв.

Окремим етапом тестування стала перевірка механізмів валідації даних, введених користувачем у вікнах конфігурації пристроїв. Було протестовано логіку перевірки коректності IP-адрес, маски підмережі, шлюзу та DNS-сервера на вкладці «Мережа», доступній для кожного пристрою з інтерфейсом. Для цього вручну вводились як допустимі, так і хибні значення. У кожному випадку застосунок реагував відповідно: у разі неправильного введення спрацьовував механізм перевірки та відображалось повідомлення про помилку у вигляді спливаючого діалогового вікна.

Тестування валідації показало, що програма не дозволяє почати симуляцію конфігурації з некоректними даними. Це дозволило переконатись, що користувач фізично не може задати недійсну IP-адресу, що унеможливлює виникнення помилок у подальшій симуляції через помилково введені параметри.

Особливої уваги заслуговує перевірка поведінки симулятора при порожніх або частково заповнених полях. У процесі тестування вводились випадки, коли заповнювалось лише одне-два поля з чотирьох необхідних. У таких ситуаціях система ігнорувала часткове збереження та виводила відповідне повідомлення про необхідність повного заповнення полів конфігурації, що підтвердило стабільність логіки валідації.

Окремо проведено тестування теоретичного модуля, а саме інтерактивного вікна з чатом на основі мовної моделі. Перевірка здійснювалась у кількох режимах: введення загальних теоретичних запитів, уточнення термінів та моделювання навчального діалогу. У ході тестування створювалися запити українською або англійською мовою, після чого система коректно формувала HTTP-запит до API OpenAI. Відповідь поверталась у форматі текстового повідомлення та відображалась у вікні діалогу. Тестування підтвердило

стабільність з’єднання з сервером, коректність формування запитів і відображення відповідей, а також правильну обробку ситуацій із помилками. Було перевірено також відповідність змісту відповідей тематиці комп’ютерних мереж, і модель демонструвала високий рівень релевантності та логічності. Таким чином, функціональність чат-вікна підтвердила свою працездатність і ефективність як інтерактивного теоретичного помічника у межах симулятора.

Проведене тестування дозволило підтвердити відповідність роботи симулятора його функціональним вимогам. Усі ключові сценарії, включно з маршрутизацією, ARP-визначенням, DNS та HTTP-взаємодією, реалізовано у відповідності до очікуваної логіки. Виявлені у процесі тестування незначні помилки були усунуті, що підвищило стабільність і надійність графічного симулятора. Процес тестування застосунку зображено на рисунку 4.1.

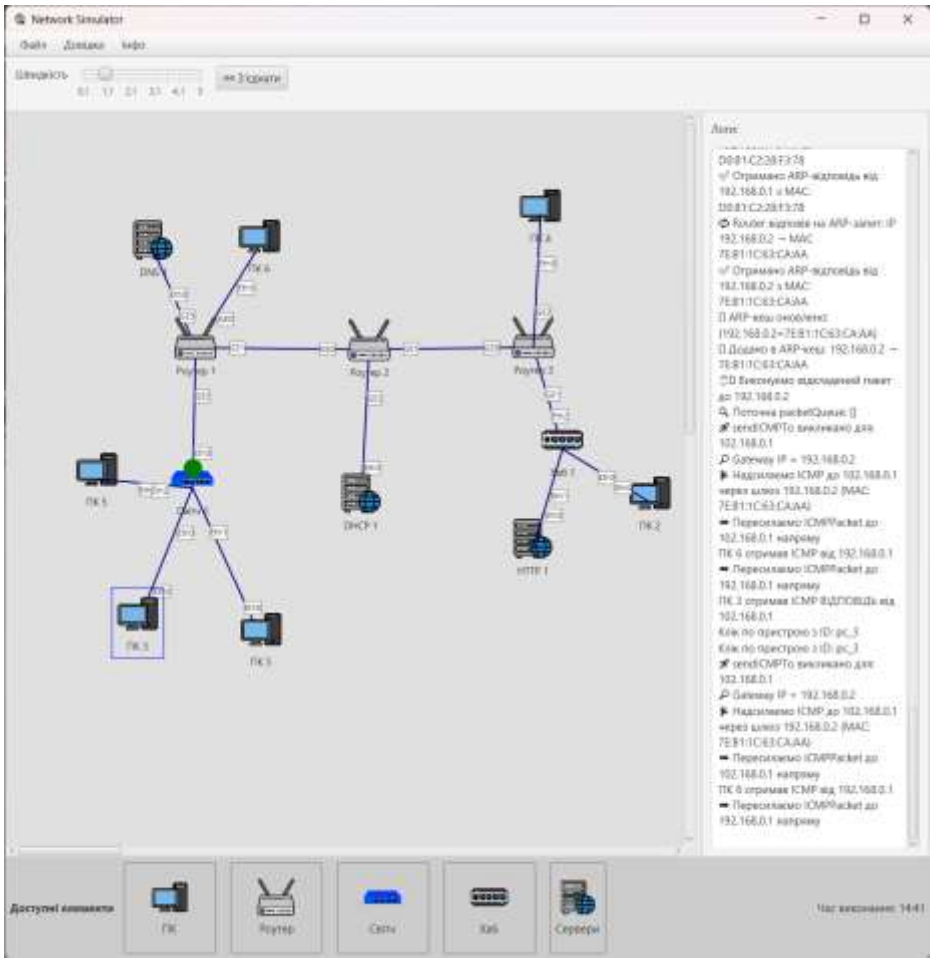


Рисунок 4.1 – Тестування застосунку

Отже, ручне тестування симулятора показало, що розроблена система повністю виконує свої функції, забезпечує інтегровану взаємодію між модулями та готова до подальшого використання в навчальних або демонстраційних цілях. Для покращення подальшої розробки та підтримки рекомендується впровадження автоматизованого тестування для перевірки змін у логіці симуляції без потреби в повторному проходженні всіх ручних сценаріїв.

4.2 Розробка інструкції користувача

Для роботи з графічним симулятором Network Simulator необхідно мати комп’ютер з операційною системою Windows, macOS або Linux, на якому встановлено Java (рекомендовано версія 17 або новіша). Графічний інтерфейс застосунку побудовано з використанням JavaFX, тому потрібна підтримка JavaFX Runtime середовища. Рекомендовано використовувати екран з роздільною здатністю від 1080×720 пікселів для повноцінного відображення усіх елементів інтерфейсу. Після запуску застосунку відкривається головне вікно, яке містить інтерактивне поле для побудови мережевої топології (див. рисунок 4.2).

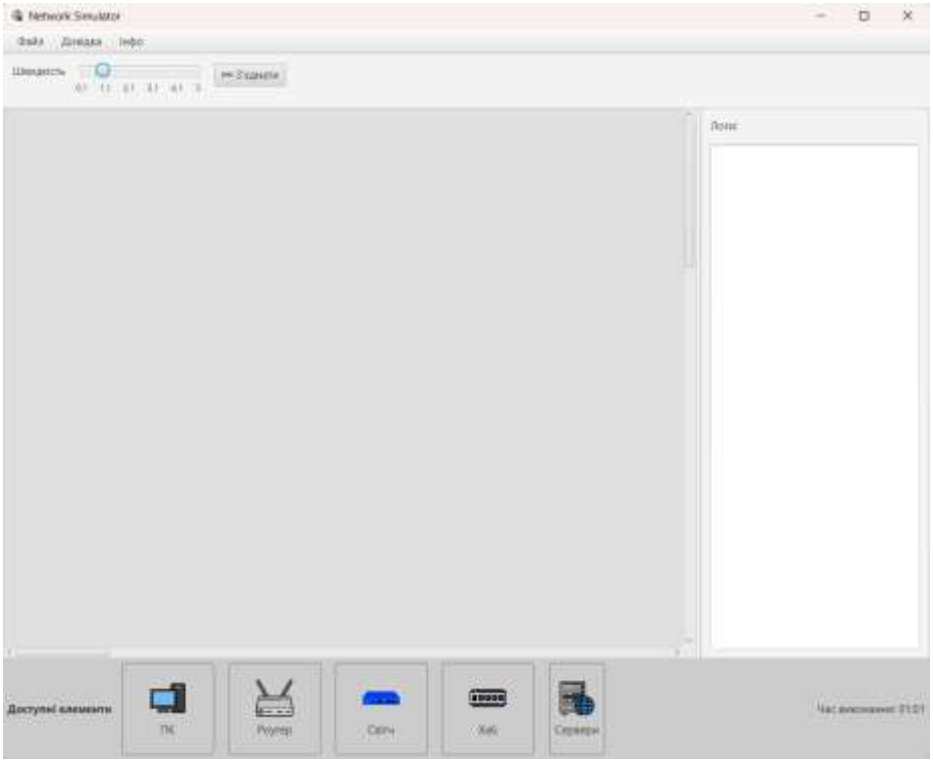


Рисунок 4.2 – Головне вікно

У верхній частині інтерфейсу розташовано панель керування, яка налаштовує швидкість симуляції (див. рисунок 4.3), зберігає проєкт у файл, відкривати наявну конфігурацію, створює нову, а також розгортає інформацію про застосунок.

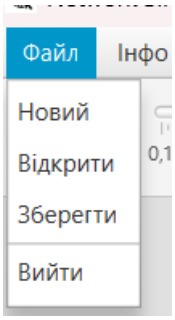


Рисунок 4.3 – Панель керування

Кнопка «Зберегти проєкт» дозволяє експортувати поточну мережеву схему, з усіма конфігураціями, у JSON-файл. Завантаження проєкту автоматично відновлює пристрої, їхні параметри та всі з’єднання.

Щоб додати пристрій, необхідно натиснути відповідну кнопку в нижній панелі «Доступні елементи», для вибору типу серверу передбачене окреме діалогове вікно з їх переліком (див. рисунок 4.4). Пристрій з’явиться у лівому верхньому кутку видимої області, його можна перетягувати за допомогою миші.

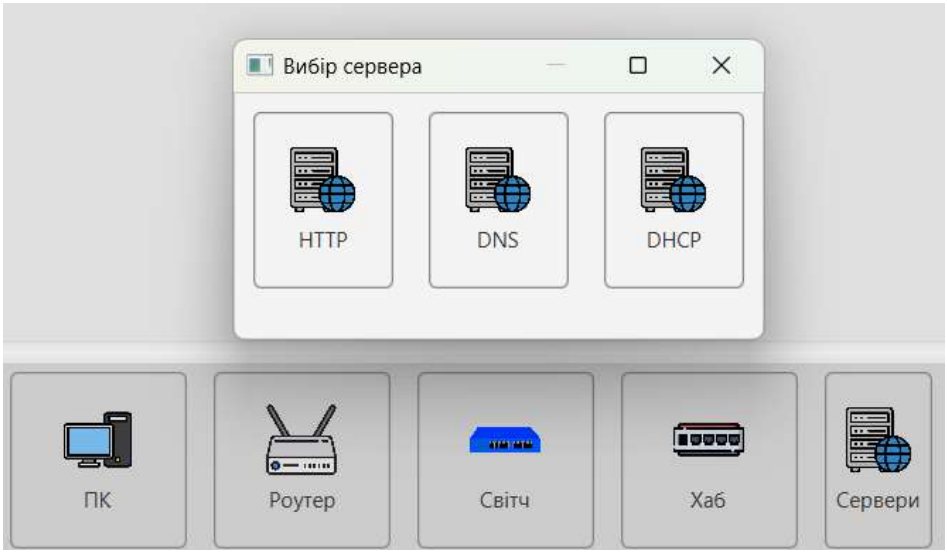


Рисунок 4.4 – Додавання пристроїв

Для створення з’єднання між пристроями потрібно обрати два необхідні пристрої, вибрати інтерфейс (див. рисунок 4.5), а потім клікнути на кнопку «З’єднати» і обрати відповідні інтерфейси для підключення. З’єднання з’явиться у вигляді лінії, яка оновлюється в реальному часі при переміщенні пристроїв.

Рисунок 4.5 – Вибір інтерфейсу на пристрої

Кожен пристрій має власне вікно налаштувань (див. рисунок 4.6), яке відкривається подвійним кліком. У вікні доступні вкладки залежно від типу пристрою. Вкладка «Мережа» дозволяє задавати IP-адресу, маску підмережі, шлюз та DNS-сервер. MAC-адреса формується автоматично. Дані зберігаються окремо для кожного інтерфейсу.

Рисунок 4.6 – Вкладка «Мережа»

Валідація введених IP-адрес і масок виконується автоматично – у разі помилки відображається повідомлення (див. рисунок 4.7), а збереження параметрів блокується.

Рисунок 4.7 – Валідація введених даних

У маршрутизаторів додатково доступна вкладка «Маршрутизація», де можна обрати режим: статична або динамічна (див. рисунок 4.8). У першому випадку вказуються маршрут, маска й адреса наступного хопа, у другому – мережі для анонсування. Дані використовуються при симуляції маршрутів.

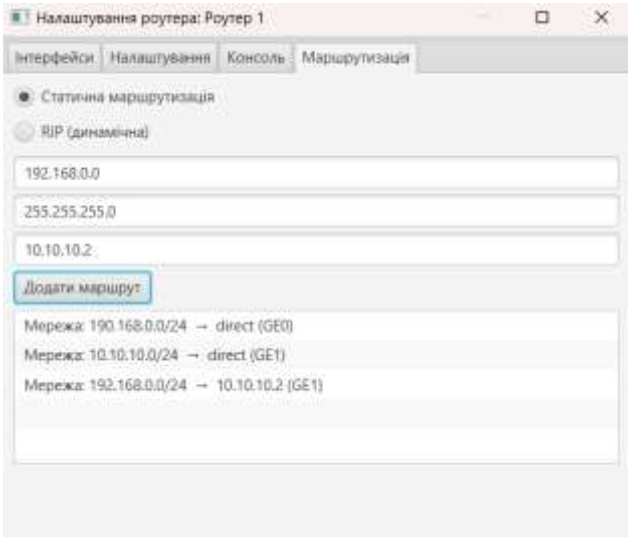


Рисунок 4.8 – Вкладка «Маршрутизація»

У DNS-сервері доступна вкладка з таблицею доменів, де можна додати відповідності імен до IP-адрес (див. рисунок 4.9). У HTTP-сервері передбачено вказування HTML-вмісту для кожного домену, який надсилатиметься у відповідь на HTTP-запит.

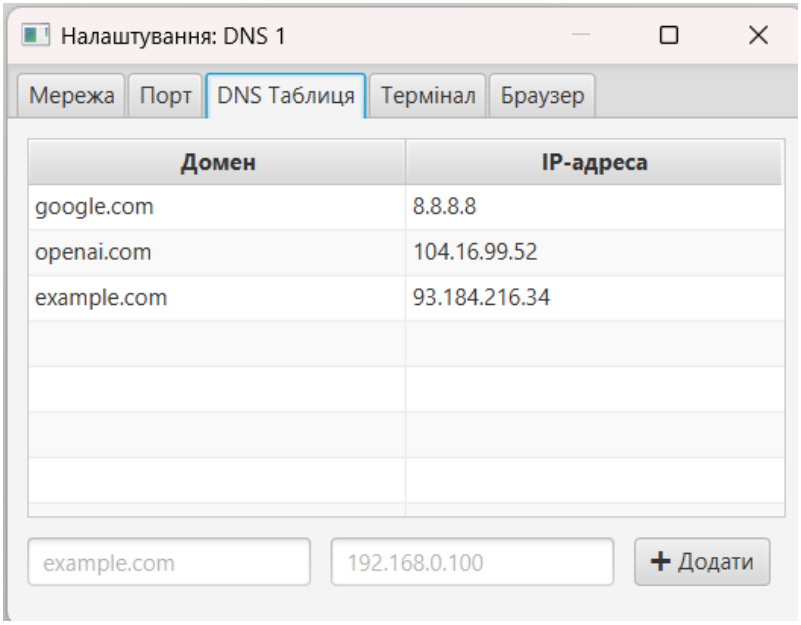


Рисунок 4.9 – Таблиця доменів

У вікнах налаштувань окремих пристроїв, зокрема персональних комп’ютерів та серверів, реалізовано додаткові вкладки: «Браузер» та «Термінал». Ці вкладки моделюють інтерфейс взаємодії користувача з мережею через знайомі аналоги – веб браузер та командну консоль.

Вкладка «Браузер» є спрощеною симуляцією текстового веб-клієнта. У верхній частині розміщено поле введення, куди користувач може ввести доменне ім’я сервера. Після натискання кнопки «Перейти» система генерує DNS-запит для визначення IP-адреси домену, а після отримання відповіді – формує HTTP-запит до відповідного сервера. У відповідь симулятор надсилає HTTP-вміст, попередньо заданий у конфігурації сервера (див. рисунок 4.10), який відображається у вікні браузера як текстова сторінка.

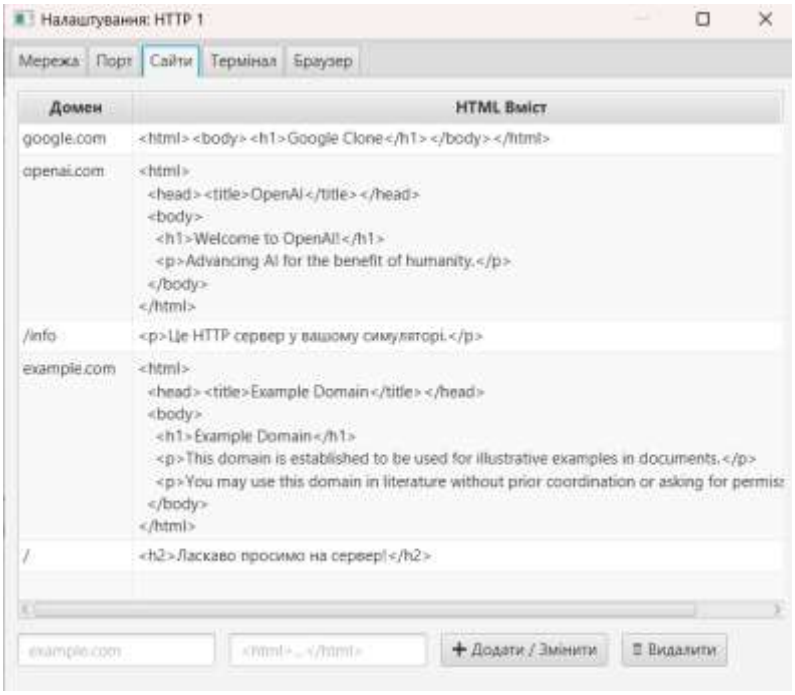


Рисунок 4.10 – Конфігурація сервера

Вкладка «Термінал» реалізована у вигляді командного рядка. Користувач може ввести симульовані мережеві команди, наприклад:

– ping – перевірка доступності пристрою через ICMP-запит;

– ipconfig – виведення IP-конфігурації пристрою (див. рисунок 4.11);

– clear – очищення терміналу від попередніх повідомлень;

– help – виведення списку підтримуваних команд із коротким описом.

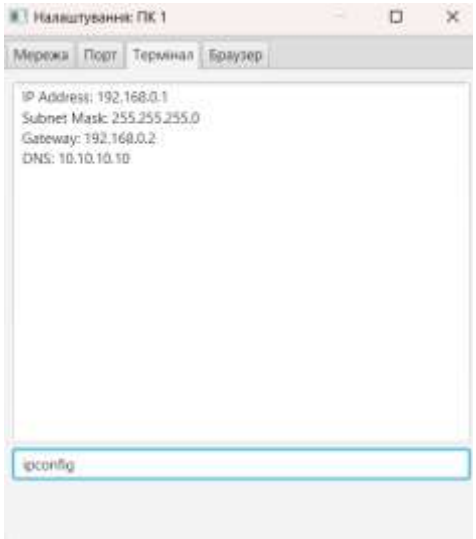


Рисунок 4.11 – Виконання ipconfig у «Термінал»

Після введення команди і натискання Enter система аналізує введення, формує відповідні події, додає їх у чергу симуляції та виконує з урахуванням всієї логіки маршрутизації, ARP-визначення та передачі. Результати команди виводяться у вигляді відповіді в текстовому полі терміналу, що дозволяє оцінити, чи команда виконана успішно, і яка відповідь отримана..

На полі з’являються анімації пакетів, які передаються між пристроями відповідно до таблиць маршрутизації, ARP або DNS-визначення. У правій частині вікна ведеться лог подій симуляції (див. рисунок 4.12), де фіксуються надсилання, прийом і обробка пакетів, зокрема тип, IP-адреси та результат доставки.

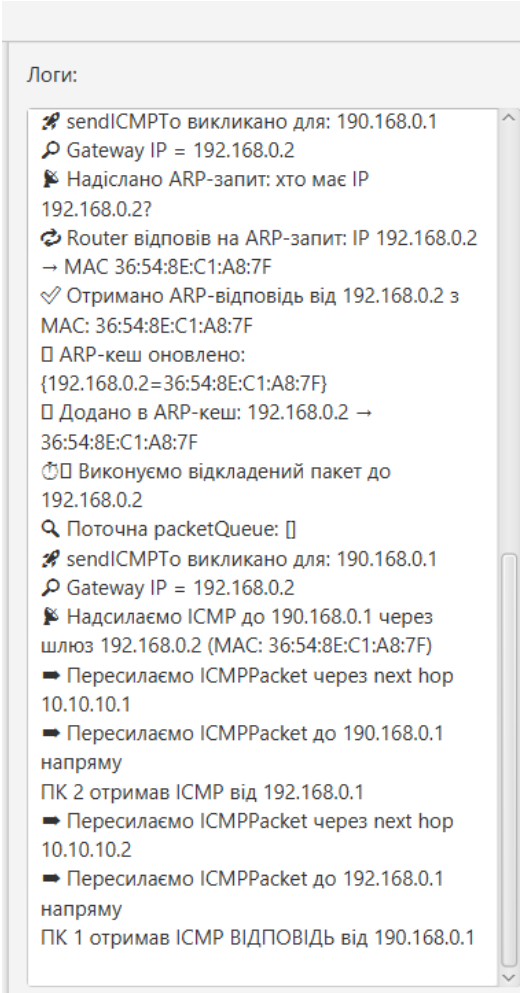


Рисунок 4.12 – Лог подій симуляції

Також реалізовано контекстне меню для кожного пристрою, яке викликається правим кліком миші по піктограмі пристрою на робочому полі.

Меню містить базові дії для керування об’єктами мережі (перейменувати, копіювати, видалити), що дозволяє швидко взаємодіяти з ними без необхідності відкривати додаткові вікна (див. рисунок 4.13).

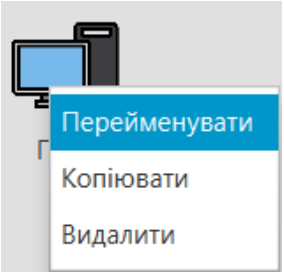


Рисунок 4.13 – Контекстне меню

У програмному застосунку реалізовано інтерактивні механізми управління пристроями за допомогою зручних дій миші та клавіатури, що підвищують комфортність роботи в середовищі.

Підтримуються такі дії:

1. Delete – натискання клавіші видаляє вибраний пристрій або з’єднання з робочого поля. Це дозволяє швидко очищати непотрібні елементи без використання контекстного меню.

2. Подвійне клацання (Double click) – відкриває вікно налаштування ім’я вибраного пристрою.

3. Ctrl + колесо прокрутки миші – змінює масштаб візуального поля, дозволяючи зручно орієнтуватися в схемі, особливо при роботі з великою кількістю пристроїв.

Також, у симуляторі реалізовано вбудований калькулятор підмережі. Користувач може ввести IP-адресу та маску підмережі, після чого система автоматично обчислює параметри мережі: адресу підмережі, широкомовну адресу, діапазон доступних IP-адрес та кількість хостів. Інтерфейс калькулятора є простим і зрозумілим, що дозволяє швидко отримувати необхідні дані при конфігурації мережевих пристроїв або перевірці коректності вручну заданих параметрів (див. рисунок 4.14).

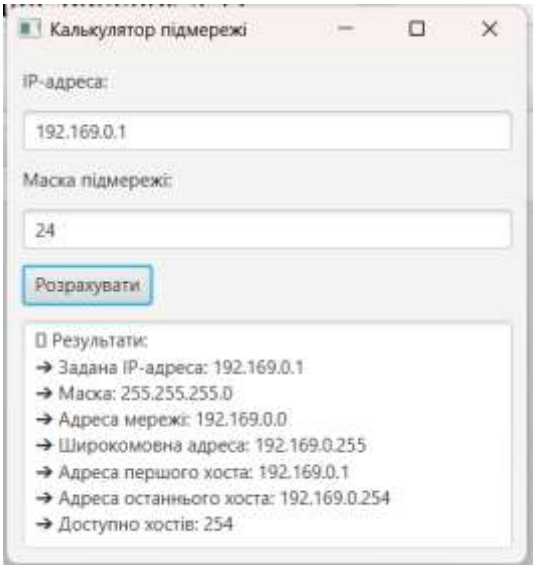


Рисунок 4.14 – Калькулятор підмережі

Крім того, для зручності навчання в симуляторі реалізовано чат-помічник на базі мовної моделі ChatGPT (див. рисунок 4.15). Він доступний у вигляді окремого вікна, де користувач може ставити запитання українською або англійською мовою.

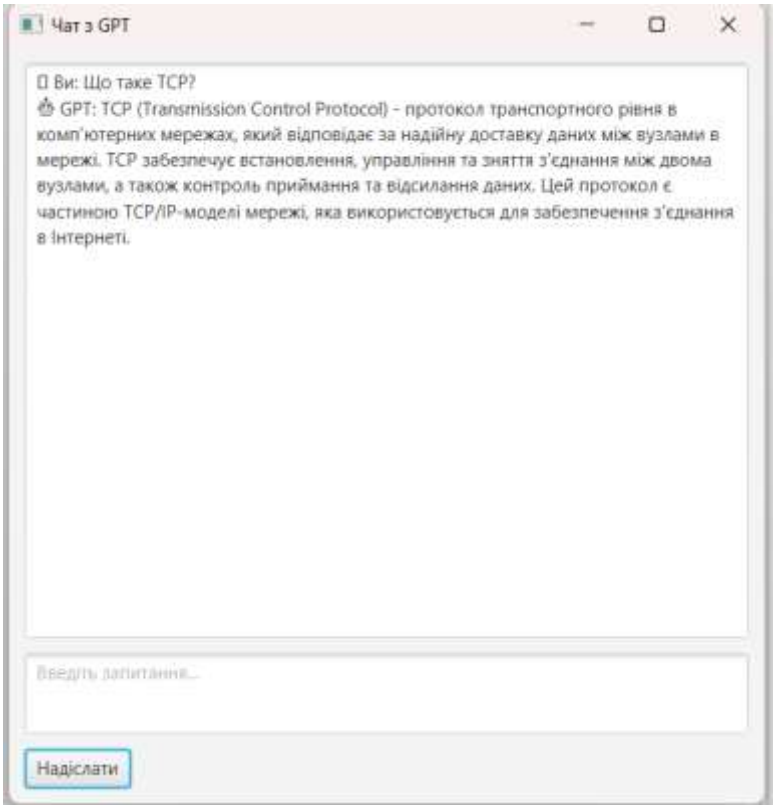


Рисунок 4.15 – Чат-помічник

Довідник містить навчальний матеріал про мережеві технології українською та англійською мовами, представлений у вигляді двох паралельних текстових блоків (див. рисунок 4.16). Вікно довідника відкривається через меню «Теорія». Обидві версії тексту доступні для одночасного перегляду, що дозволяє вивчати термінологію у двох мовах паралельно. Також реалізовано пошук по обох текстах — користувач може ввести ключове слово, і всі знайдені збіги автоматично підсвітяться в обох мовних версіях.

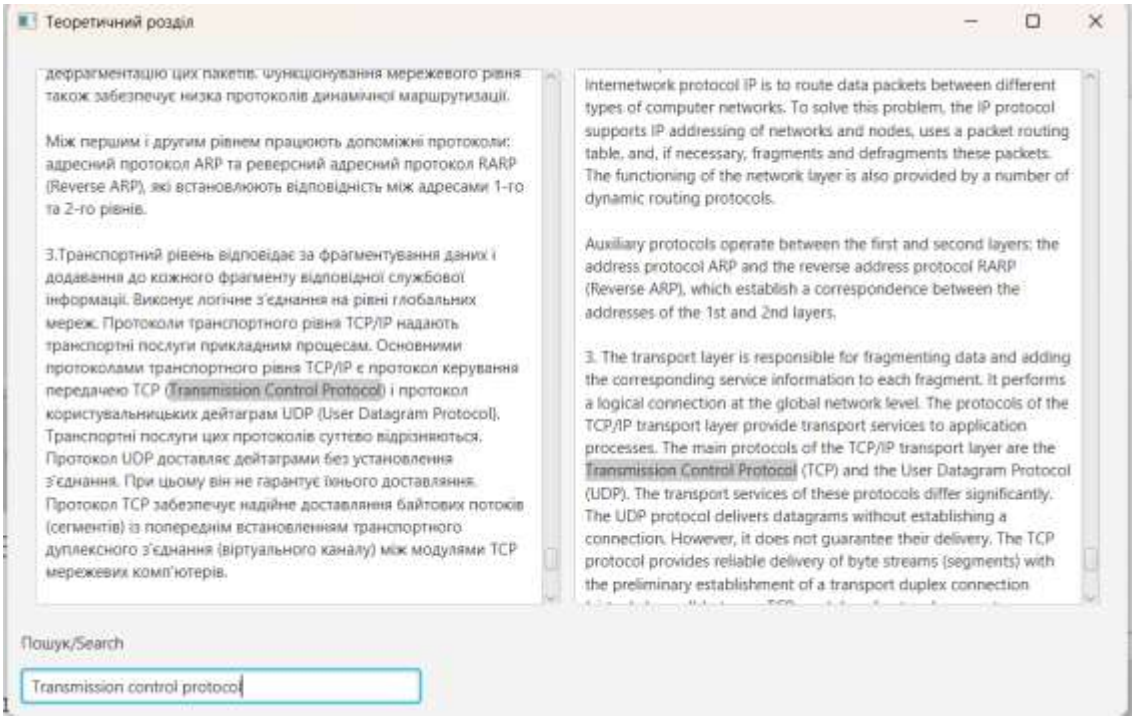


Рисунок 4.16 – Довідник

Загалом, програмна система Network Simulator є повноцінним графічним середовищем для моделювання комп’ютерних мереж, яке забезпечує широкий спектр функціональності – від побудови топології до симуляції мережевих протоколів. Користувач може створювати мережі, налаштовувати параметри пристроїв, спостерігати за передачею пакетів і аналізувати їхній рух у режимі реального часу.

4.3 Висновки

Отже, було проведено тестування програмної системи моделювання мережевих топологій та розроблено інструкцію користувача. Завдяки використанню ручного тестування вдалося оцінити коректність роботи основних функцій системи з точки зору кінцевого користувача, виявлено недоліки юзабіліті та перевірити стабільність при різних сценаріях моделювання.

У процесі тестування було перевірено створення пристроїв, з’єднання між ними, конфігурацію мережевих параметрів, взаємодію з вікнами налаштувань, роботу протоколів, а також механізми збереження і завантаження проєктів. Окрему увагу приділено коректності роботи терміналу та браузера, валідності введених параметрів, реакції на некоректні команди та обробці виключень. Перевірялась також робота масштабування, контекстного меню та дій користувача при маніпуляції з об’єктами на полотні.

Тестування показало, що всі ключові компоненти симулятора взаємодіють належним чином, а результати симуляції відповідають очікуваній поведінці мережі. Виявлені під час тестування недоліки були усунені, що дозволило покращити стабільність і користувацький досвід.

Інструкція користувача розроблена для забезпечення повного охоплення можливостей застосунку. Описано механіку побудови топологій, конфігурацію пристроїв, використання терміналу, браузера, контекстного меню, а також реалізовані гарячі клавіші. Структура інструкції дозволяє швидко знайти потрібну інформацію для роботи з симулятором.

ВИСНОВКИ

У роботі розроблено методи та засоби реалізації графічного симулятора для візуалізації та дослідження 7-рівневої моделі OSI та стеку протоколів TCP/IP.

Здійснено аналіз сучасних програмних засобів для моделювання мережевих технологій, визначено їхні функціональні обмеження та обґрунтовано доцільність створення нового симулятора. На основі порівняльного аналізу було сформульовано цілі, завдання та вимоги до графічного симулятора.

У результаті аналізу технологічної бази було обґрунтовано використання мови програмування Java, фреймворку JavaFX для реалізації графічного інтерфейсу та бібліотеки Gson для збереження та обміну даними у форматі JSON.

У межах виконання проєкту реалізовано такі модулі:

– проаналізовано засоби та методи для реалізації симулятора мережевих технологій;

– розроблено розробити архітектуру та модель роботи симулятора;

– розроблено модуль симуляції, що забезпечує обробку подій і логіку роботи протоколів;

– розроблено модуль побудови мережевої топології для створення пристроїв, інтерфейсів і з’єднань між ними;

– реалізовано модуль конфігурації пристроїв, що надає можливість налаштування службових параметрів;

– створено модуль збереження та завантаження проєктів, який дозволяє серіалізувати стан мережі у форматі JSON;

– розроблено модуль для отримання супровідної теоретичної інформації;

– розроблено керівництво користувача, яке забезпечить експлуатацію та подальший розвиток додатку;

– реалізовано графічний інтерфейс програмного застосунку.

Також, удосконалено метод візуалізації та симуляції процесів взаємодії рівнів моделі OSI та стеку TCP/IP. Подальший розвиток отримали інструменти для отримання та візуалізації теоретичного матеріалу щодо 7-рівневої моделі OSI та стеку протоколів TCP/IP в графічному симуляторі.

Результати тестування підтвердили працездатність симулятора, коректність реалізації основних протоколів і зручність взаємодії користувача з інтерфейсом.

Бакалаврську кваліфікаційну роботу оформлено відповідно до методичних вказівок [32].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Computer Networks: веб-сайт. URL: https://www.sciencedirect.com /journal/computer-networks (дата звернення: 26.03.2025).

2. Інформаційні системи : веб-сайт. URL: https://step.org.ua/konspekt /inform/tema1 (дата звернення: 30.03.2025).

3. Multi-level network architecture: веб-сайт. URL: https://www.researchgate.net/figure/Multi-level-network-architecture-for-Internet-based-content-distri-bution_fig1_299377684 (дата звернення: 02.04.2025).

4. Модель OSI: веб-сайт. URL: https://dou.ua/forums/topic/46215/ (дата звернення: 03.04.2025).

5. TCP/IP protocols: веб-сайт. URL: https://www.ibm.com /docs/en/aix/7.3.0?topic=protocol-tcpip-protocols (дата звернення: 07.04.2025).

6. What is Abstraction: веб-сайт. URL: https://pwskills.com/blog/what-is-abstraction-in-oops-definition-types-and-advantages/ (дата звернення: 08.04.2025).

7. Josh Waitzkin. The Art of Learning: monograph. – New York: Scribner, 2007. 265 p.

8. Network Simulator: веб-сайт. URL: https://www.sciencedirect.com /topics/computer-science/network-simulator (дата звернення: 09.04.2025).

9. Сулим М.Ю. JavaFX як інструмент розробки графічних симуляторів / М. Ю. Сулим, Д. І. Катєльніков // Матеріали I Міжнародної науково-практичної конференції «Innovations in Science: From Theoretical Foundations to Practical Impact». – Антверпен, Бельгія, 2025. – С. 153-155 (дата звернення: 12.04.2025).

10. What is Artificial Intelligence?: веб-сайт. URL: https://cloud.google.com/learn/what-is-artificial-intelligence (дата звернення: 15.04.2025).

11. Сулим М.Ю. Інкапсуляція та декапсуляція: як дані мандрують мережею / Мирослав Юрійович Сулим, Денис Іванович Катєльніков // Матеріали LIV Всеукраїнської науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії. – Вінниця, 2025, URL:

https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/ 24447/ 20155 (дата звернення: 15.04.2025).

12. Як працює DNS: веб-сайт. URL: https://hostiq.ua/blog/ukr/how-does-dns-work/ (дата звернення: 18.04.2025).

13. Інтерактивність: веб-сайт. URL: https://dictionary.cambridge.org /dictionary/english/interactivity (дата звернення: 18.04.2025).

14. IP-адреса: веб-сайт. URL: https://trium.com.ua/43-shcho-take-ip-adresa-i-dlia-choho-vona-potribna (дата звернення: 20.04.2025).

15. Cisco Packet Tracer: веб-сайт. URL: https://www.netacad.com/cisco-packet-tracer (дата звернення: 21.04.2025).

16. Graphical Network Simulator 3: веб-сайт. URL: https://www.gns3.com (дата звернення: 25.04.2025).

17. Emulated Virtual Environment Next Generation: веб-сайт. URL: https://www.eve-ng.net (дата звернення: 05.05.2025).

18. Boson NetSim: веб-сайт. URL: https://www.boson.com/netsim-cisco-network-simulator?srsltid=AfmBOopwwwuTjViBoBw8aN7liI95MLYA27b-nk8CpnHmwVERY5Wnacot (дата звернення: 06.05.2025).

19. Монолітна архітектура ПЗ : веб-сайт. URL: https://qalight.ua/baza-znaniy/shho-take-monolitna-arhitektura/ (дата звернення: 10.05.2025).

20. Java : веб-сайт. URL: https://www.java.com/en/ (дата звернення: 10.05.2025).

21. What is a MAC address? : веб-сайт. URL: https://www.portnox.com/cybersecurity-101/mac-address (дата звернення: 11.05.2025).

22. ARP cache: What is it and how can it help you? : веб-сайт. URL: https://petri.com/csc_arp_cache/ (дата звернення: 14.05.2025).

23. Routing Information Protocol: веб-сайт. URL: https://www.techtarget.com/searchnetworking/definition/Routing-Information-Protocol (дата звернення: 14.05.2025).

24. HTML - Hypertext Markup Language: веб-сайт. URL: https://www.freelancer.com.ua/what-is-html (дата звернення: 17.05.2025).

25. Working with JSON: веб-сайт. URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Scripting/ JSON (дата звернення: 18.05.2025).

26. Об’єктно-Орієнтоване Програмування (ООП). Пояснюємо чотири основні принципи: веб-сайт. URL: https://career.softserveinc.com/uk-ua/stories/what-is-object-oriented-programming-oop-explaining-four-major-principles (дата звернення: 18.05.2025).

27. Gson - Quick Guide: веб-сайт. URL: https://www.tutorialspoint .com/gson/gson_quick_guide.htm (дата звернення: 19.05.2025).

28. Internet Control Message Protocol Definition: веб-сайт. URL: https://www.fortinet.com/resources/cyberglossary/internet-control-message-protocol-icmp (дата звернення: 20.05.2025).

29. ChatGpt: веб-сайт. URL: https://openai.com/index/chatgpt/ (дата звернення: 21.05.2025).

30. What Is an API (Application Programming Interface)?: веб-сайт. URL: https://www.oracle.com/ua/cloud/cloud-native/api-management/what-is-api/ (дата звернення: 21.05.2025).

31. HTTP-client: веб-сайт. URL: https://symfony.com.ua /doc/current/http_client.html (дата звернення: 22.05.2025).

32. Романюк О. Н., Черноволик Г. О. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 121 «Інженерія програмного забезпечення». Вінниця: ВНТУ, 2022. 52 с.

ДОДАТКИ

ДОДАТОК А

(обов'язковий)

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

«25» 03 2025 р.

Технічне завдання

на бакалаврську кваліфікаційну роботу «Розробка графічного симулятора для візуалізації та дослідження 7-рівневої моделі OSI і стеку протоколів TCP/IP з використанням фреймворку JavaFX» за спеціальністю 121 Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:

к.т.н., доцент Д. І. Катєльніков

24.03 2025 р.

Виконав:

студент гр. 5ПІ-21б М. Ю. Сулим

«24» 03 2025 р.

Вінниця – 2025 року

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Розробка графічного симулятора для візуалізації та дослідження 7-рівневої моделі OSI і стеку протоколів TCP/IP з використанням фреймворку JavaFX».

Галузь застосування – системи комп’ютерних мереж.

2. Підстава для розробки.

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №97 від «20» березня 2025 р. ректора по ВНТУ про закріплення тем БКР.

3. Мета та призначення розробки.

Метою роботи є покращення візуалізації та опису процесів мережевих технологій, з використанням засобів інформаційних технологій, шляхом розробки програмних засобів для графічного симулятора для візуалізації та дослідження функціонування 7-рівневої моделі OSI та стеку протоколів TCP/IP.

Призначення роботи – розробка методів і засобів симуляції комп’ютерної мережі.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БКР:

1. Сулим М.Ю. Інкапсуляція та декапсуляція: як дані мандрують мережею / Мирослав Юрійович Сулим, Денис Іванович Катєльніков // Матеріали LIV Всеукраїнської науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії. – Вінниця, 2025, URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2025/paper/view/ 24447/ 20155.

2. Сулим М.Ю. JavaFX як інструмент розробки графічних симуляторів / М. Ю. Сулим, Д. І. Катєльніков // Матеріали І Міжнародної науково-практичної

конференції «Innovations in Science: From Theoretical Foundations to Practical Impact». – Антверпен, Бельгія, 2025. – С. 153-155.

3. Модель OSI: веб-сайт. URL: https://dou.ua/forums/topic/46215/.

4. TCP/IP protocols: веб-сайт. URL: https://www.ibm.com /docs/en/aix/7.3.0?topic=protocol-tcpip-protocols.

5. Об’єктно-Орієнтоване Програмування (ООП). Пояснюємо чотири основні принципи: веб-сайт. URL: https://career.softserveinc.com/uk-ua/stories/what-is-object-oriented-programming-oop-explaining-four-major-principles.

5. Технічні вимоги

Підтримка виведення елементів мережі у векторному вигляді з можливістю масштабування, перетягування та динамічного оновлення; підтримка Unicode; графічний режим – повна кольорова палітра; розмір полотна – адаптивний до вікна користувача. Вхідні дані – конфігурації пристроїв, координати об'єктів, параметри інтерфейсів, запити користувача. Вихідні дані – візуалізація мережі та пакетів, відповіді чат-бота, оброблені події, зміни стану мережі.

6. Конструктивні вимоги.

Інтерфейс повинен бути естетично привабливим, логічно структурованим та зручним. Текстова та графічна документація повинна відповідати стандартам України.

7. Перелік технічної документації, що пред’являється по закінченню робіт:

– пояснювальна записка до БКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз моделей мережевої взаємодії та вибір напрямів досліджень	25.03.25 – 08.04.25
2	Розробка моделі та алгоритмів графічного симулятора	09.04.25 – 16.04.25
3	Розробка модулів реалізації симулятора	17.04.25 – 03.05.25
4	Тестування програмного застосунку	04.05.25 – 19.05.25
5	Оформлення матеріалів до захисту БКР	20.05.25 – 30.05.25

10. Порядок контролю та прийняття.

Виконання етапів бакалаврської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Захист бакалаврської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

ДОДАТОК Б

(обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка графічного симулятора для візуалізації та дослідження 7-рівневої моделі OSI і стеку протоколів TCP/IP з використанням фреймворку JavaFX

Тип роботи: БКР

Підрозділ: кафедра програмного забезпечення, ФІТКІ.

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 2,0 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Катєльніков Д. І.
(підпис)

Здобувач _____ Сулим М. Ю.
(підпис)

ДОДАТОК В

(довідниковий)

Лістинг програмного коду графічного симулятора для візуалізації та дослідження 7-рівневої моделі OSI і стеку протоколів TCP/IP

MainApp.java

```
package com.schedule.kursovyproject;

import com.schedule.kursovyproject.model.*;
import com.schedule.kursovyproject.save.*;
import com.schedule.kursovyproject.util.*;
import com.schedule.kursovyproject.simulation.*;
import javafx.scene.control.TextInputDialog;
import javafx.scene.input.KeyCode;
import javafx.scene.input.MouseButton;
import javafx.scene.shape.Rectangle;
import javafx.scene.paint.Color;
import javafx.geometry.Bounds;
import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.application.Application;
import javafx.event.EventHandler;
import javafx.scene.Group;
import javafx.geometry.Insets;
import javafx.geometry.Orientation;
import javafx.geometry.Pos;
import javafx.scene.Node;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.image.Image;
import javafx.application.Platform;
import java.io.IOException;
import java.io.InputStream;
import java.nio.charset.StandardCharsets;
import java.util.List;
import java.util.ArrayList;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.FileNotFoundException;
import java.util.stream.Collectors;
import javafx.scene.image.ImageView;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.*;
import javafx.stage.FileChooser;
import javafx.stage.Modality;
```

```
import javafx.stage.Stage;
import javafx.util.Duration;
import javafx.scene.input.ContextMenuEvent;
import java.io.File;
import java.time.LocalTime;
import java.time.format.DateTimeFormatter;
import java.util.HashMap;
import java.util.Map;
import com.schedule.kursovyproject.ui.*;
public class MainApp extends Application {

    private static MainApp instance;
    private Pane simulationPane;
    private ScrollPane scrollPane;
    private double selectionStartX;
    private double selectionStartY;
    private int secondsElapsed = 0;
    private double scale = 1.0;
    private Label timerLabel;
    private TextArea logArea;
    private Timeline timer;
    private final java.util.List<NetworkDevice> selectedDevices = new java.util.ArrayList<>();
    private static final List<NetworkDevice> allDevices = new ArrayList<>();
    private final Rectangle selectionRect = new Rectangle();
    private int deviceCounter = 1;
    public MainApp() {
        instance = this;
    }
    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Network Simulator");
        primaryStage.getIcons().add(
                new
Image(getClass().getResourceAsStream("/com/schedule/kursovyproject/images/logo.png"))
        );
        Menu fileMenu = new Menu("Файл");
        MenuItem newItem = new MenuItem("Новий");
        MenuItem openItem = new MenuItem("Відкрити");
        newItem.setOnAction(e -> {
            Alert confirmDialog = new Alert(Alert.AlertType.CONFIRMATION);
            confirmDialog.setTitle("Новий проєкт");
            confirmDialog.setHeaderText("Бажаєте зберегти поточний проєкт перед створенням
нового?");
            confirmDialog.setContentText("Зміни буде втрачено без збереження.");
            ButtonType saveButton = new ButtonType("Зберегти");
            ButtonType discardButton = new ButtonType("Не зберігати");
            ButtonType cancelButton = new ButtonType("Скасувати",
ButtonBar.ButtonData.CANCEL_CLOSE);
            confirmDialog.getButtonTypes().setAll(saveButton, discardButton, cancelButton);

            confirmDialog.showAndWait().ifPresent(response -> {
                if (response == saveButton) {
```

```
                FileChooser fileChooser = new FileChooser();
                fileChooser.setTitle("Зберегти проєкт");
                fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("Файли
проєкту (*.json)", "*.json"));
                File file = fileChooser.showSaveDialog(primaryStage);
                if (file != null) {
                    if (!file.getName().toLowerCase().endsWith(".json")) {
                        file = new File(file.getAbsolutePath() + ".json");
                    }
                    SavedProject project = ProjectBuilder.createFromCurrentState(allDevices);
                    ProjectIO.saveToFile(project, file.getAbsolutePath());
                }
                clearSimulationAndResetTimer();
            } else if (response == discardButton) {
                clearSimulationAndResetTimer();
            }
        });
    });
    openItem.setOnAction(e -> {
        FileChooser fileChooser = new FileChooser();
        fileChooser.setTitle("Відкрити проєкт");
        fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("JSON файли",
"*.json"));
        File file = fileChooser.showOpenDialog(primaryStage);

        if (file != null) {
            SavedProject project = ProjectIO.loadFromFile(file.getAbsolutePath());
            if (project != null) {
                clearSimulation();
                buildFromProject(project);
            }
        }
    });

    MenuItem saveItem = new MenuItem("Зберегти");
    saveItem.setOnAction(e -> {
        FileChooser fileChooser = new FileChooser();
        fileChooser.setTitle("Зберегти проєкт");
        fileChooser.getExtensionFilters().add(
                new FileChooser.ExtensionFilter("Файли проєкту (*.json)", "*.json")
        );
        File file = fileChooser.showSaveDialog(primaryStage);
        if (file != null) {
            if (!file.getName().toLowerCase().endsWith(".json")) {
                file = new File(file.getAbsolutePath() + ".json");
            }
            SavedProject project = ProjectBuilder.createFromCurrentState(allDevices);
            ProjectIO.saveToFile(project, file.getAbsolutePath());
        }
    });
    MenuItem exitItem = new MenuItem("Вийти");
    exitItem.setOnAction(e -> primaryStage.close());
```

```
        fileMenu.getItems().addAll(newItem, openItem, saveItem, new SeparatorMenuItem(),
exitItem);
        Menu referenceMenu = new Menu("Довідка");
        MenuItem subnetCalcItem = new MenuItem("Калькулятор");
        subnetCalcItem.setOnAction(e -> SubnetCalculatorWindow.open());
        MenuItem theoryItem = new MenuItem("Теорія");
        theoryItem.setOnAction(e -> {
            try {
                String ukrText =
loadTextFromResources("com/schedule/kursovyproject/theory_ukr.txt");
                String engText =
loadTextFromResources("com/schedule/kursovyproject/theory_eng.txt");
                TextArea ukrArea = new TextArea(ukrText);
                ukrArea.setWrapText(true);
                ukrArea.setEditable(false);
                TextArea engArea = new TextArea(engText);
                engArea.setWrapText(true);
                engArea.setEditable(false);
                Label search = new Label("Пошук/Search");
                TextField searchField = new TextField();
                searchField.setMaxWidth(300);
                searchField.textProperty().addListener((obs, oldVal, newVal) -> {
                    highlightMatch(ukrArea, newVal);
                    highlightMatch(engArea, newVal);
                });
                HBox textBox = new HBox(10, ukrArea, engArea);
                textBox.setPadding(new Insets(10));
                ukrArea.setPrefWidth(1200);
                engArea.setPrefWidth(1200);
                ukrArea.setPrefHeight(1500);
                engArea.setPrefHeight(1500);
                VBox layout = new VBox(10, textBox, search, searchField);
                layout.setPadding(new Insets(10));
                Stage theoryStage = new Stage();
                theoryStage.setTitle("Теоретичний розділ");
                theoryStage.setScene(new Scene(layout, 1200, 900));
                theoryStage.show();
                Platform.runLater(() -> {
                    ScrollPane ukrScroll = (ScrollPane) ukrArea.lookup(".scroll-pane");
                    ScrollPane engScroll = (ScrollPane) engArea.lookup(".scroll-pane");
                    if (ukrScroll != null && engScroll != null) {
                        ukrScroll.vvalueProperty().bindBidirectional(engScroll.vvalueProperty());
                    }
                });
            } catch (Exception ex) {
                MainApp.log("❌ Не вдалося завантажити теорію: " + ex.getMessage());
            }
        });
        MenuItem chatItem = new MenuItem("AI асистент");
        chatItem.setOnAction(e -> ChatWindow.open());
        referenceMenu.getItems().addAll(subnetCalcItem, theoryItem, chatItem);
        Menu infoMenu = new Menu("Інфо");
```

```
        MenuItem aboutItem = new MenuItem("Про застосунок");
        aboutItem.setOnAction(e -> {
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setTitle("Про застосунок");
            alert.setHeaderText("Симулятор OSI TCP/IP");
            alert.setContentText("Цей застосунок створено для моделювання мережевої взаємодії
за моделлю OSI.");
            alert.showAndWait();
        });
        infoMenu.getItems().add(aboutItem);
        MenuBar menuBar = new MenuBar(fileMenu, referenceMenu, infoMenu);
        simulationPane = new Pane();
        simulationPane.setStyle("-fx-background-color: #e0e0e0;");
        Group zoomGroup = new Group(simulationPane);
        scrollPane = new ScrollPane(zoomGroup);
        selectionRect.setStroke(Color.BLUE);
        selectionRect.setStrokeWidth(1);
        selectionRect.setFill(Color.web("blue", 0.1));
        selectionRect.setVisible(false);
        simulationPane.getChildren().add(selectionRect);

        simulationPane.setOnMousePressed(e -> {
            if (e.isPrimaryButtonDown()) {
                selectionStartX = e.getX();
                selectionStartY = e.getY();
                selectionRect.setX(selectionStartX);
                selectionRect.setY(selectionStartY);
                selectionRect.setWidth(0);
                selectionRect.setHeight(0);
                selectionRect.setVisible(true);
            }
        });
        simulationPane.setOnMouseDragged(e -> {
            if (e.isPrimaryButtonDown()) {
                double x = e.getX();
                double y = e.getY();
                selectionRect.setX(Math.min(x, selectionStartX));
                selectionRect.setY(Math.min(y, selectionStartY));
                selectionRect.setWidth(Math.abs(x - selectionStartX));
                selectionRect.setHeight(Math.abs(y - selectionStartY));
            }
        });

        simulationPane.setOnMouseReleased(e -> {
            if (selectionRect.isVisible()) {
                Bounds bounds = selectionRect.getBoundsInParent();
                for (Node node : simulationPane.getChildren()) {
                    if (node instanceof NetworkDevice) {
                        NetworkDevice device = (NetworkDevice) node;
                        if (device.getBoundsInParent().intersects(bounds)) {
                            device.setStyle("-fx-border-color: blue; -fx-border-width: 1;");
                            if (!selectedDevices.contains(device)) {
```

```
                    selectedDevices.add(device);
  }  } } }
      selectionRect.setVisible(false);
   }
});
zoomGroup.setOnScroll(event -> {
   if (event.isControlDown()) {
      double delta = 0.1;
      if (event.getDeltaY() < 0) {
         scale -= delta;
      } else {
         scale += delta;
      }
      scale = Math.max(0.5, Math.min(scale, 3));
      simulationPane.setScaleX(scale);
      simulationPane.setScaleY(scale);
      event.consume();
   }
});
scrollPane.setPannable(true);
scrollPane.setHbarPolicy(ScrollPane.ScrollBarPolicy.AS_NEEDED);
scrollPane.setVbarPolicy(ScrollPane.ScrollBarPolicy.AS_NEEDED);
simulationPane.setMinSize(5000, 2000);
timerLabel = new Label("Час: 00:00");
startTimer();
Label elementsLabel = new Label("Доступні елементи");
elementsLabel.setStyle("-fx-font-weight: bold;");
HBox elementIconsBox = new HBox(15);
elementIconsBox.setAlignment(Pos.CENTER_LEFT);
elementIconsBox.getChildren().addAll(
      createButton("ПК", "pc.png"),
      createButton("Роутер", "router.png"),
      createButton("Світч", "switch.png"),
      createButton("Хаб", "hub.png"),
      createServerButton()
);
Region spacer = new Region();
HBox.setHgrow(spacer, Priority.ALWAYS);
HBox bottomHBox = new HBox(10);
bottomHBox.setPadding(new Insets(5));
bottomHBox.setStyle("-fx-background-color: #cccccc;");
bottomHBox.setAlignment(Pos.CENTER_LEFT);
bottomHBox.getChildren().addAll(elementsLabel, elementIconsBox, spacer, timerLabel);

logArea = new TextArea();
logArea.setEditable(false);
logArea.setWrapText(true);
logArea.setPrefWidth(300);
VBox.setVgrow(logArea, Priority.ALWAYS);

VBox logPanel = new VBox(10, new Label("Логи:"), logArea);
logPanel.setPadding(new Insets(10));
```

```
logPanel.setPrefWidth(320);
SplitPane splitPane = new SplitPane();
splitPane.setOrientation(Orientation.HORIZONTAL);
splitPane.getItems().addAll(scrollPane, logPanel);
splitPane.setDividerPositions(0.75);

HBox controlPanel = new HBox(10);
controlPanel.setPadding(new Insets(10));
Slider speedSlider = new Slider(0.1, 5.0, 1.0);
speedSlider.setOrientation(Orientation.HORIZONTAL);
speedSlider.setShowTickLabels(true);
speedSlider.setShowTickMarks(true);
speedSlider.setMajorTickUnit(0.5);
controlPanel.getChildren().addAll(new Label("Швидкість"), speedSlider);
Button connectBtn = new Button("🔗 З’єднати");
controlPanel.getChildren().add(connectBtn);
VBox topPanel = new VBox();
topPanel.getChildren().addAll(menuBar, controlPanel);
connectBtn.setOnAction(e -> {
   if (selectedDevices.size() == 2) {
      NetworkDevice d1 = selectedDevices.get(0);
      NetworkDevice d2 = selectedDevices.get(1);
      Dialog<LinkType> dialog = new Dialog<>();
      dialog.setTitle("Тип з'єднання");
      ComboBox<Interface> ifaceBox1 = new ComboBox<>();
      ifaceBox1.getItems().addAll(d1.getInterfaces());
      ifaceBox1.setValue(d1.getAvailableInterface());
      ComboBox<Interface> ifaceBox2 = new ComboBox<>();
      ifaceBox2.getItems().addAll(d2.getInterfaces());
      ifaceBox2.setValue(d2.getAvailableInterface());
      ComboBox<LinkType> typeBox = new ComboBox<>();
      typeBox.getItems().addAll(LinkType.values());
      typeBox.setValue(LinkType.ETHERNET);
      typeBox.setCellFactory(cb -> new ListCell<>() {
         @Override
         protected void updateItem(LinkType item, boolean empty) {
            super.updateItem(item, empty);
            if (item == null || empty) {
               setText(null);
            } else {
               String label;
               switch (item) {
                  case ETHERNET:
                     label = "Ethernet cable";
                     break;
                  case CROSSOVER:
                     label = "Copper cross-over";
                     break;
                  case FIBER:
                     label = "Fiber optic cable";
                     break;
                  case WIRELESS:
```

```
                        label = "Wireless link";
                        break;
                    default:
                        label = item.name();
                }
                setText(label);
            }
        }
    });
    typeBox.setButtonCell(typeBox.getCellFactory().call(null));
    VBox dialogContent = new VBox(10,
        new Label("Оберіть тип з'єднання:"), typeBox,
        new Label("Інтерфейс " + d1.getDeviceName() + ":"), ifaceBox1,
        new Label("Інтерфейс " + d2.getDeviceName() + ":"), ifaceBox2
    );
    dialogContent.setPadding(new Insets(10));
    dialog.getDialogPane().setContent(dialogContent);
    dialog.getDialogPane().getButtonTypes().addAll(ButtonType.OK,
ButtonType.CANCEL);
    dialog.setResultConverter(btn -> btn == ButtonType.OK ? typeBox.getValue() : null);
    dialog.showAndWait().ifPresent(type -> {
        Interface i1 = ifaceBox1.getValue();
        Interface i2 = ifaceBox2.getValue();
        if (i1 == null || i2 == null) {
            Alert alert = new Alert(Alert.AlertType.WARNING);
            alert.setTitle("З’єднання");
            alert.setHeaderText(null);
            alert.setContentText("Інтерфейси не вибрані.");
            alert.showAndWait();
            return;
        }

        for (Interface iface : List.of(i1, i2)) {
            if (iface.isConnected()) {
                Link oldLink = iface.getLink();
                oldLink.getFrom().disconnect();
                oldLink.getTo().disconnect();
                simulationPane.getChildren().remove(oldLink.getLine());
            }
        }
        try {
            Link link = new Link(i1, i2, type, simulationPane);
            i1.getDevice().getLinks().add(link);
            i2.getDevice().getLinks().add(link);
        } catch (IllegalArgumentException ex) {
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("Помилка з'єднання");
            alert.setHeaderText("Несумісні інтерфейси");
            alert.setContentText(ex.getMessage());
            alert.showAndWait();
        }
        for (NetworkDevice d : selectedDevices) {
```

```
                d.setStyle("-fx-border-color: transparent; -fx-border-width: 1;");
            }
            selectedDevices.clear();
        });
    } else {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("З'єднання");
        alert.setHeaderText(null);
        alert.setContentText("Виділи рівно 2 пристрої для з'єднання.");
        alert.showAndWait();
    }
});
BorderPane root = new BorderPane();
Scene scene = new Scene(root, 1000, 1000);
primaryStage.setScene(scene);
scene.setOnKeyPressed(event -> {
    if (event.getCode() == KeyCode.DELETE) {
        for (NetworkDevice device : new ArrayList<>(selectedDevices)) {
            for (Interface iface : device.getInterfaces()) {
                iface.removeLinkIfConnected();
            }
            simulationPane.getChildren().remove(device);
            MainApp.log("Видалено "+ device.getDeviceName());
            allDevices.remove(device);
        }
        selectedDevices.clear();
    }
});
root.setRight(logPanel);
root.setTop(topPanel);
root.setCenter(scrollPane);
root.setBottom(bottomHBox);
root.setCenter(splitPane);
primaryStage.show();
}

private String generateUniqueId(String prefix) {
    return prefix + "_" + (deviceCounter++);
}
public static void log(String message) {
    if (instance == null || instance.logArea == null) return;
    Platform.runLater(() -> instance.logArea.appendText(message + "\n"));
}
private VBox createButton(String name, String imageFile) {
    Image image = new
Image(getClass().getResourceAsStream("/com/schedule/kursovyproject/images/" + imageFile));
    ImageView imageView = new ImageView(image);
    imageView.setFitWidth(40);
    imageView.setFitHeight(40);

    Label label = new Label(name);
    label.setAlignment(Pos.CENTER);
```

```
        VBox box = new VBox(5, imageView, label);
        box.setAlignment(Pos.CENTER);
        box.setPadding(new Insets(5));
        box.setStyle("-fx-border-color: gray; -fx-border-radius: 4;");
        box.setPrefWidth(100);
        box.setPrefHeight(100);

        box.setOnMouseClicked(e -> {
            double x = scrollPane.getHvalue() * (simulationPane.getWidth() -
scrollPane.getViewportBounds().getWidth());
            double y = scrollPane.getVvalue() * (simulationPane.getHeight() -
scrollPane.getViewportBounds().getHeight());
            spawnDevice(name, image, x + 20, y + 20);
        });
        return box;
    }
    private void spawnDevice(String name, Image image, double x, double y) {
        String id;
        NetworkDevice device;
        switch (name) {
            case "ПК":
                id = generateUniqueDeviceId("pc");
                device = new PCDevice(id, generateNameById(id, name), image);
                break;
            case "Роутер":
                id = generateUniqueDeviceId("router");
                device = new RouterDevice(id, generateNameById(id, name), image);
                break;
            case "Світч":
                id = generateUniqueDeviceId("switch");
                device = new SwitchDevice(id, generateNameById(id, name), image);
                break;
            case "Хаб":
                id = generateUniqueDeviceId("hub");
                device = new HubDevice(id, generateNameById(id, name), image);
                break;
            case "DNS":
                id = generateUniqueDeviceId("dns");
                device = new DNSServerDevice(id, generateNameById(id, name), image);
                break;
            case "HTTP":
                id = generateUniqueDeviceId("http");
                device = new HTTPServerDevice(id, generateNameById(id, name), image);
                break;
            case "DHCP":
                id = generateUniqueDeviceId("dhcp");
                device = new DHCPServerDevice(id, generateNameById(id, name), image);
                break;
            default:
                MainApp.log("❌ Невідомий тип пристрою: " + name);
                return;
```

```
        }
        device.setLayoutX(x);
        device.setLayoutY(y);
        allDevices.add(device);
        simulationPane.getChildren().add(device);
        enableDragging(device);
        enableSelection(device, device.getLabel());
        MainApp.log("☑ Створено пристрій: " + name + " [" + id + "]");
    }
    private VBox createServerButton() {
        ImageView imageView = new ImageView(new
Image(getClass().getResourceAsStream("/com/schedule/kursovyproject/images/server.png")));
        imageView.setFitWidth(40);
        imageView.setFitHeight(40);
        Label label = new Label("Сервери");
        label.setAlignment(Pos.CENTER);
        VBox box = new VBox(5, imageView, label);
        box.setAlignment(Pos.CENTER);
        box.setPadding(new Insets(5));
        box.setStyle("-fx-border-color: gray; -fx-border-radius: 4;");
        box.setOnMouseClicked(event -> openServerGrid());
        return box;
    }

    private void openServerGrid() {
        Stage popupStage = new Stage();
        popupStage.initModality(Modality.APPLICATION_MODAL);
        popupStage.setTitle("Вибір сервера");
        GridPane grid = new GridPane();
        grid.setPadding(new Insets(10));
        grid.setHgap(20);
        grid.setVgap(20);

        String[][] servers = {
                {"HTTP", "server.png"},
                {"DNS", "server.png"},
                {"DHCP", "server.png"}
        };

        for (int i = 0; i < servers.length; i++) {
            VBox serverBox = createButton(servers[i][0], servers[i][1]);
            int col = i % 4;
            int row = i / 4;
            grid.add(serverBox, col, row);
        }
        ScrollPane scrollPane = new ScrollPane(grid);
        scrollPane.setFitToWidth(true);

        Scene scene = new Scene(scrollPane, 300, 140);
        popupStage.setScene(scene);
        popupStage.show();
    }
```

```
    private void enableDragging(NetworkDevice node) {
        final Delta dragDelta = new Delta();
        node.setOnMousePressed(mouseEvent -> {
            dragDelta.x = mouseEvent.getX();
            dragDelta.y = mouseEvent.getY();
            node.toFront();
            mouseEvent.consume();
        });

        node.setOnMouseDragged(mouseEvent -> {
            double dx = mouseEvent.getX() - dragDelta.x;
            double dy = mouseEvent.getY() - dragDelta.y;
            List<NetworkDevice> targets = selectedDevices.contains(node)
                    ? selectedDevices
                    : List.of(node);

            for (NetworkDevice device : targets) {
                double newX = device.getLayoutX() + dx;
                double newY = device.getLayoutY() + dy;

                newX = Math.max(0, Math.min(newX, simulationPane.getWidth() - device.getWidth()));
                newY = Math.max(0, Math.min(newY, simulationPane.getHeight() -
device.getHeight()));
                device.setLayoutX(newX);
                device.setLayoutY(newY);
                for (Link link : device.getLinks()) {
                    link.updateLinePosition();
                }
            }
            mouseEvent.consume();
        });
    }
    private static class Delta {
        double x, y;
    }

    private void startTimer() {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("mm:ss");
        timer = new Timeline(new KeyFrame(Duration.seconds(1), e -> {
            secondsElapsed++;
            LocalTime time = LocalTime.ofSecondOfDay(secondsElapsed);
            timerLabel.setText("Час виконання: " + time.format(formatter));
        }));
        timer.setCycleCount(Timeline.INDEFINITE);
        timer.play();
    }
    private String generateUniqueDeviceId(String typePrefix) {
        long count = MainApp.getAllDevices().stream()
                .filter(d -> d.getDeviceId().startsWith(typePrefix))
                .count();
        return typePrefix.toLowerCase() + "_" + (count + 1);
    }
```

```
private String generateNameById(String id, String name) {
    String[] parts = id.split("_");
    if (parts.length == 2) {
        String number = parts[1];
        return name + " " + number;
    }
    return name;
}
private void enableSelection(NetworkDevice device, Label nameLabel) {
    EventHandler<MouseEvent> clickHandler = event -> {
        MainApp.log("Клік по пристрою з ID: " + device.getDeviceId());
        if (event.getClickCount() == 2) {
            if (device instanceof DHCPServerDevice) {
                DHCPServerSettingsWindow.open((DHCPServerDevice) device);
            } else if (device instanceof HTTPServerDevice) {
                HTTPServerSettingsWindow.open((HTTPServerDevice) device);
            } else if (device instanceof DNSServerDevice) {
                DNSServerSettingsWindow.open((DNSServerDevice) device);
            } else if (device instanceof PCDevice) {
                PCSettingsWindow.open((PCDevice) device);
            } else if (device instanceof RouterDevice) {
                RouterSettingsWindow.open((RouterDevice) device);
            } else if (device instanceof HubDevice) {
                HubSettingsWindow.open((HubDevice) device);
            } else if (device instanceof SwitchDevice) {
                SwitchSettingsWindow.open((SwitchDevice) device);
            } else {
                openSettingsWindow(nameLabel);
            }
        } else if (event.getButton() == MouseButton.PRIMARY) {
            boolean selected = device.getStyle().contains("-fx-border-color: blue");

            if (event.isControlDown()) {
                if (selected) {
                    device.setStyle("-fx-border-color: transparent; -fx-border-width: 1;");
                    selectedDevices.remove(device);
                } else {
                    device.setStyle("-fx-border-color: blue; -fx-border-width: 1;");
                    selectedDevices.add(device);
                }
            } else {
                for (NetworkDevice d : new ArrayList<>(selectedDevices)) {
                    d.setStyle("-fx-border-color: transparent; -fx-border-width: 1;");
                }
                selectedDevices.clear();
                device.setStyle("-fx-border-color: blue; -fx-border-width: 1;");
                selectedDevices.add(device);
            }}
        event.consume();
    };
    device.setOnMouseClicked(clickHandler);
    device.getLabel().setOnMouseClicked(clickHandler);
```

```
if (!device.getChildren().isEmpty() && device.getChildren().get(0) instanceof Node) {
    device.getChildren().get(0).setOnMouseClicked(clickHandler);
}
device.setOnKeyPressed(event -> {
    if (event.getCode() == javafx.scene.input.KeyCode.DELETE) {
        for (Interface iface : device.getInterfaces()) {
            iface.removeLinkIfConnected();
        }
        ((Pane) device.getParent()).getChildren().remove(device);
        selectedDevices.remove(device);
    }
});
ContextMenu contextMenu = new ContextMenu();
MenuItem deleteItem = new MenuItem("Видалити");
deleteItem.setOnAction(e -> {
    if (selectedDevices.size() > 1) {
        for (NetworkDevice d : new ArrayList<>(selectedDevices)) {
            for (Link link : new ArrayList<>(d.getLinks())) {
                simulationPane.getChildren().remove(link.getLine());
                NetworkDevice other = (link.getFrom().getDevice() == d)
                        ? link.getTo().getDevice()
                        : link.getFrom().getDevice();
                other.getLinks().remove(link);
                d.getLinks().remove(link);
            }
            simulationPane.getChildren().remove(d);
        }
        selectedDevices.clear();
    } else {
        for (Interface iface : device.getInterfaces()) {
            iface.removeLinkIfConnected();
        }
        simulationPane.getChildren().remove(device);
        selectedDevices.remove(device);
        MainApp.log("Видалено "+ device.getDeviceName());
        allDevices.remove(device);
    }
});
MenuItem cloneItem = new MenuItem("Копіювати");
cloneItem.setOnAction(e -> {
    NetworkDevice clone;
    String baseName = device.getDeviceName() + " (копія)";
    String newId = generateUniqueId("clone");
    Image icon = device.getIcon().getImage();
    if (device instanceof PCDevice) {
        PCDevice original = (PCDevice) device;
        PCDevice copy = new PCDevice(newId, baseName, icon);
        copy.setDeviceName(baseName);
        for (Interface iface : original.getInterfaces()) {
            copy.getInterfaces().add(new Interface(iface.getName(), iface.getType(), copy));
        }
        clone = copy;
```

```
        } else if (device instanceof RouterDevice) {
            clone = new RouterDevice(newId, baseName, icon);
        } else if (device instanceof SwitchDevice) {
            clone = new SwitchDevice(newId, baseName, icon);
        } else if (device instanceof HubDevice) {
            clone = new HubDevice(newId, baseName, icon);
        } else if (device instanceof DNSServerDevice) {
            clone = new DNSServerDevice(newId, baseName, icon);
        } else {
            return;
        }
        clone.setLayoutX(device.getLayoutX() + 30);
        clone.setLayoutY(device.getLayoutY() + 30);
        simulationPane.getChildren().add(clone);
        enableDragging(clone);
        enableSelection(clone, clone.getLabel());
    });

    MenuItem renameItem = new MenuItem("Перейменувати");
    renameItem.setOnAction(e -> {
        TextInputDialog dialog = new TextInputDialog(device.getDeviceName());
        dialog.setTitle("Перейменування пристрою");
        dialog.setHeaderText(null);
        dialog.setContentText("Нова назва (до 30 символів):");
        dialog.showAndWait().ifPresent(newName -> {
            if (newName.length() > 30) {
                Alert alert = new Alert(Alert.AlertType.WARNING);
                alert.setTitle("Помилка");
                alert.setHeaderText(null);
                alert.setContentText("Назва пристрою не може перевищувати 30 символів.");
                alert.showAndWait();
            } else {
                device.setDeviceName(newName);
            }
        });
    });
    contextMenu.getItems().addAll(renameItem, cloneItem, deleteItem);
    device.setOnContextMenuRequested((ContextMenuEvent event) -> {
        contextMenu.show(device, event.getScreenX(), event.getScreenY());
    });
}
private void openSettingsWindow(Label labelRef) {
    Stage dialog = new Stage();
    dialog.initModality(Modality.APPLICATION_MODAL);
    dialog.setTitle("Налаштування: " + labelRef.getText());

    VBox layout = new VBox(10);
    layout.setPadding(new Insets(10));

    TextField nameField = new TextField(labelRef.getText());
    layout.getChildren().addAll(new Label("Ім’я пристрою:"), nameField);
```

```
        Button saveBtn = new Button("Зберегти");
        saveBtn.setOnAction(e -> {
            labelRef.setText(nameField.getText());
            dialog.close();
        });

        Button closeBtn = new Button("Закрити");
        closeBtn.setOnAction(e -> dialog.close());

        HBox buttons = new HBox(10, saveBtn, closeBtn);
        layout.getChildren().add(buttons);

        Scene scene = new Scene(layout, 300, 150);
        dialog.setScene(scene);
        dialog.show();
    }
    private void clearSimulationAndResetTimer() {
        clearSimulation();
        secondsElapsed = 0;
        timerLabel.setText("Час: 00:00");
    }
    public static List<NetworkDevice> getAllDevices() {
        return allDevices;
    }
    public static void main(String[] args) {
        launch(args);
    }

    public void clearSimulation() {
        simulationPane.getChildren().clear();
        allDevices.clear();
        selectedDevices.clear();
    }

    public void buildFromProject(SavedProject project) {
        Map<String, NetworkDevice> idToDevice = new HashMap<>();
        for (SavedDevice saved : project.devices) {
            NetworkDevice device;

            switch (saved.type.toLowerCase()) {
                case "pc":
                    device = new PCDevice(saved.id, saved.name, DeviceIcons.getPC(), saved.layoutX,
saved.layoutY, saved.interfaces);
                    break;
                case "router":
                    RouterDevice router = new RouterDevice(saved.id, saved.name,
DeviceIcons.getRouter(), saved.layoutX, saved.layoutY, saved.interfaces);
                    router.setUseRelay(saved.useRelay);
                    router.setRelayTargetIp(saved.relayTargetIp);

                    if (saved.routes != null) {
                        for (SavedRoute sr : saved.routes) {
```

```
                    Interface out = router.getInterfaces().stream()
                            .filter(i -> i.getName().equals(sr.outInterfaceName))
                            .findFirst().orElse(null);
                    router.addRoute(new RoutingTableEntry(sr.network, sr.mask, sr.nextHop, out));
                }
            }

            device = router;
            break;
        case "switch":
            device = new SwitchDevice(saved.id, saved.name, DeviceIcons.getSwitch(),
saved.layoutX, saved.layoutY, saved.interfaces);
            break;
        case "hub":
            device = new HubDevice(saved.id, saved.name, DeviceIcons.getHub(), saved.layoutX,
saved.layoutY, saved.interfaces);
            break;
        case "dnsserver":
            device = new DNSServerDevice(saved.id, saved.name, DeviceIcons.getServer(),
saved.layoutX, saved.layoutY, saved.interfaces);
            break;
        case "httpserver":
            device = new HTTPServerDevice(saved.id, saved.name, DeviceIcons.getServer(),
saved.layoutX, saved.layoutY, saved.interfaces);
            break;
        case "dhcpserver":
            DHCPServerDevice dhcp = new DHCPServerDevice(saved.id, saved.name,
DeviceIcons.getServer(), saved.layoutX, saved.layoutY, saved.interfaces);
            dhcp.setNetworkPrefix(saved.networkPrefix);
            dhcp.setSubnetMask(saved.subnetMask);
            dhcp.setGateway(saved.gateway);
            dhcp.setDnsServerIp(saved.dnsServerIp);
            dhcp.setNextHostId(saved.nextHostId);
            device = dhcp;
            break;
        default:
            MainApp.log("❌ Невідомий тип пристрою: " + saved.type);
            continue;
    }
    device.forceSetMacAddress(saved.macAddress);

    if (!simulationPane.getChildren().contains(device)) {
        simulationPane.getChildren().add(device);
    }
    allDevices.add(device);
    idToDevice.put(saved.id, device);

    enableDragging(device);
    enableSelection(device, device.getLabel());
}
for (SavedLink link : project.links) {
    NetworkDevice fromDevice = idToDevice.get(link.fromDeviceId);
```

```
            NetworkDevice toDevice = idToDevice.get(link.toDeviceId);
            if (fromDevice == null || toDevice == null) continue;
            Interface fromIface = fromDevice.getInterfaces().stream()
                    .filter(i -> i.getName().equals(link.fromInterfaceName)).findFirst().orElse(null);
            Interface toIface = toDevice.getInterfaces().stream()
                    .filter(i -> i.getName().equals(link.toInterfaceName)).findFirst().orElse(null);
            if (fromIface != null && toIface != null) {
                Link l = new Link(fromIface, toIface, LinkType.valueOf(link.linkType), simulationPane);
                if (!simulationPane.getChildren().contains(l.getLine())) {
                    simulationPane.getChildren().add(0, l.getLine());
                }
            }
        }
    }

    public static String loadTextFromResources(String relativePath) {
        try (InputStream input = MainApp.class.getResourceAsStream("/" + relativePath)) {
            if (input == null) {
                return "❌ Файл не знайдено: " + relativePath;
            }
            return new BufferedReader(new InputStreamReader(input))
                    .lines()
                    .collect(Collectors.joining("\n"));
        } catch (Exception e) {
            return "❌ Помилка при читанні файлу: " + e.getMessage();
        }
    }

    public static void highlightMatch(TextArea area, String word) {
        if (word == null || word.isEmpty()) return;

        String content = area.getText().toLowerCase();
        int index = content.indexOf(word.toLowerCase());

        if (index >= 0) {
            area.selectRange(index, index + word.length());
        } else {
            area.deselect();
        }
    }
}
```

ДОДАТОК Г

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ГРАФІЧНОГО СИМУЛЯТОРА ДЛЯ ВІЗУАЛІЗАЦІЇ ТА ДОСЛІДЖЕННЯ 7-РІВНЕВОЇ МОДЕЛІ OSI І СТЕКУ ПРОТОКОЛІВ TCP/IP З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ JAVAFX

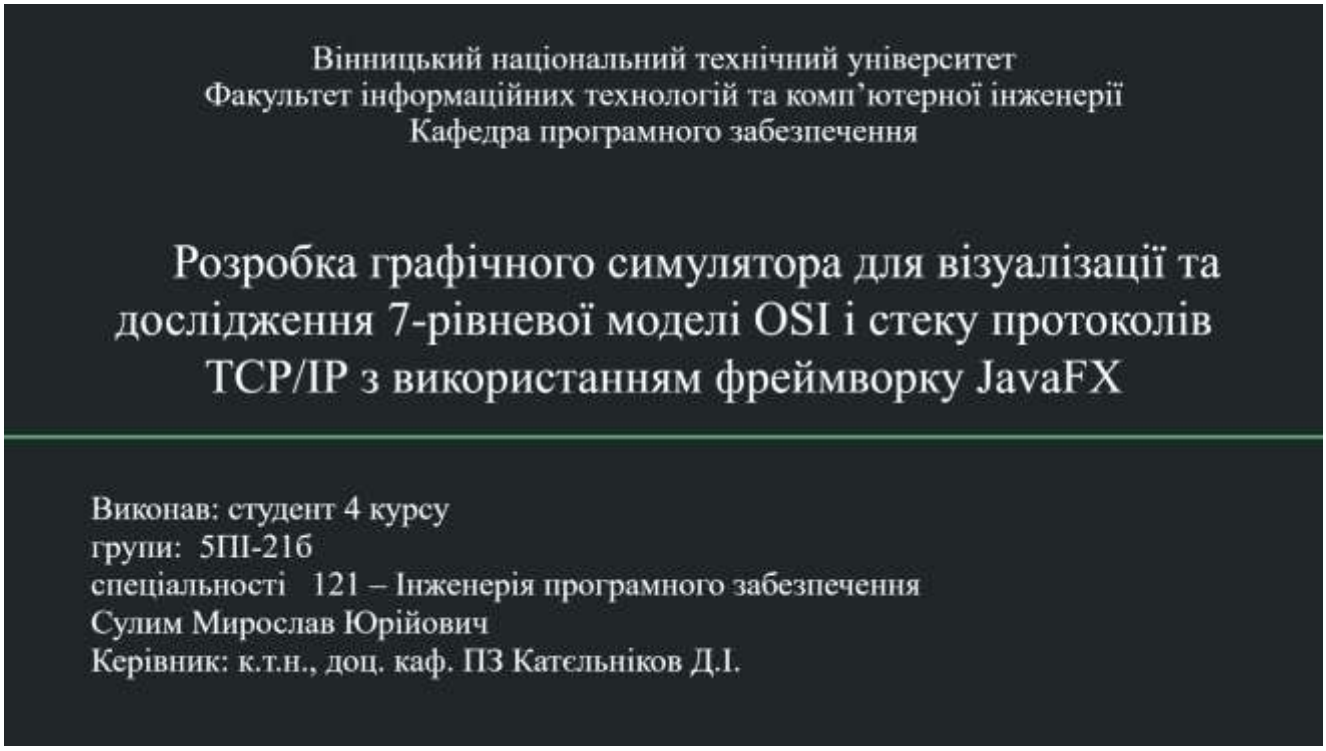


Рисунок Г.1 – Титульний аркуш

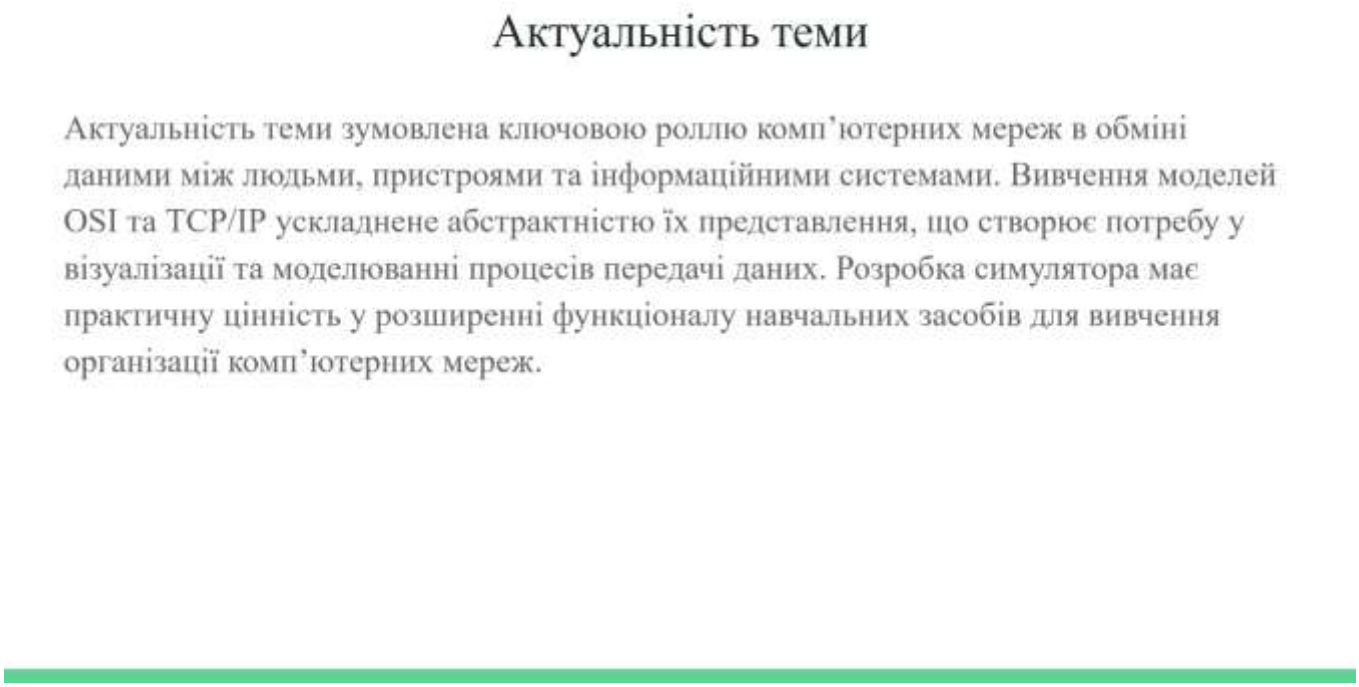


Рисунок Г.2 – Актуальність теми

Рисунок Г.3 – Мета, предмет, об'єкт кваліфікаційної роботи

Наукова новизна

Подальшого розвитку отримав метод візуалізації та симуляції процесів взаємодії рівнів моделі OSI та стеку TCP/IP, який, на відміну від існуючих, використовує абстрактні класи та рівні наслідування, що дозволяє інтерпретувати мережеві процеси у наочній формі з можливістю інтерактивної зміни параметрів і спостереження результатів у режимі реального часу.

Подальший розвиток отримали метод побудови мережевої топології, який, на відміну від існуючих, має вбудований чат для використання штучного інтелекту, що дозволяє вчасно отримувати коментарі щодо правильного виконання операцій з'єднання мережевих компонент.

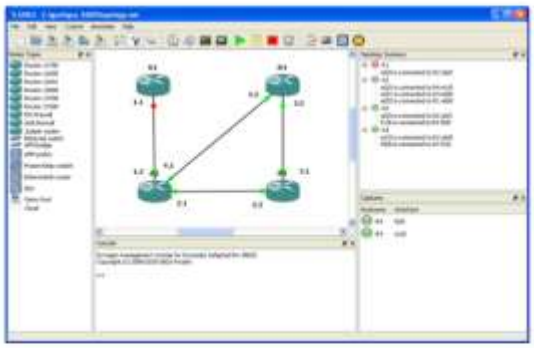
Рисунок Г.4 – Наукова новизна кваліфікаційної роботи

Задачі бакалаврської кваліфікаційної роботи

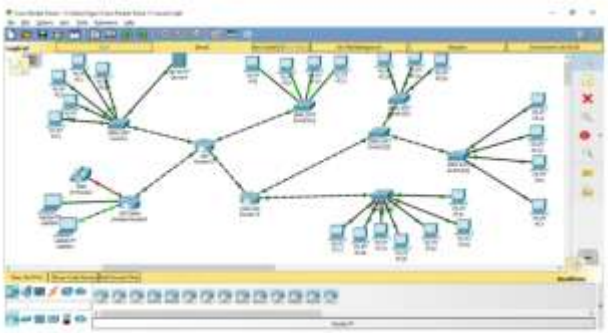
- проаналізувати засоби та методи для реалізації симулятора мережевих технологій;
- розробити архітектуру та модель роботи симулятора;
- створити алгоритми взаємодії рівнів моделі OSI та TCP/IP;
- розробити графічний інтерфейс застосунку;
- реалізувати модульну структуру симулятора, що забезпечить інтерактивність і наочність навчання;
- провести тестування і налагодження розробленого програмного продукту.

Рисунок Г.5 – Задачі бакалаврської кваліфікаційної роботи

Аналоги



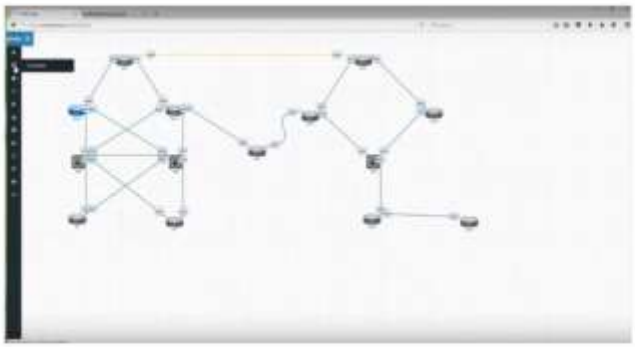
Cisco packet tracer



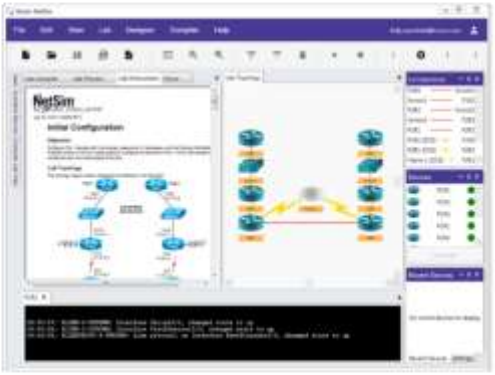
Graphical Network Simulator 3

Рисунок Г.6 – Аналоги

Аналоги



Emulated Virtual Environment Next Generation



Boson NetSim

Рисунок Г.7 – Аналоги

Порівняльний аналіз аналогів

Критерій	Packet Tracer	GNS3	EVE-NG	Boson NetSim	Network Simulator
Мережева логіка	+	+	+	+	+
Передача даних	+	+	-	-	+
Конфігурація пристроїв	+	-	+	+	+
Гарячі клавіші	+	-	+	+	+
Відкритість до модифікацій	-	+	+	-	+
Теоретичний модуль	+	-	-	+	+
Загалом	5	3	4	4	6

Рисунок Г.8 – Порівняльний аналіз аналогів

Діаграма діяльності користувача з симулятором

Рисунок Г.9 – Діаграма діяльності

Блок-схема симуляції HTTP-запиту

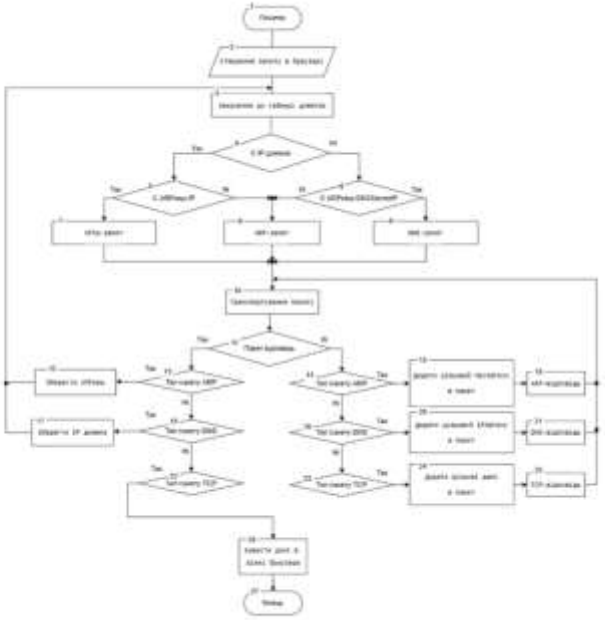


Рисунок Г.10 – Блок-схема симуляції HTTP-запиту

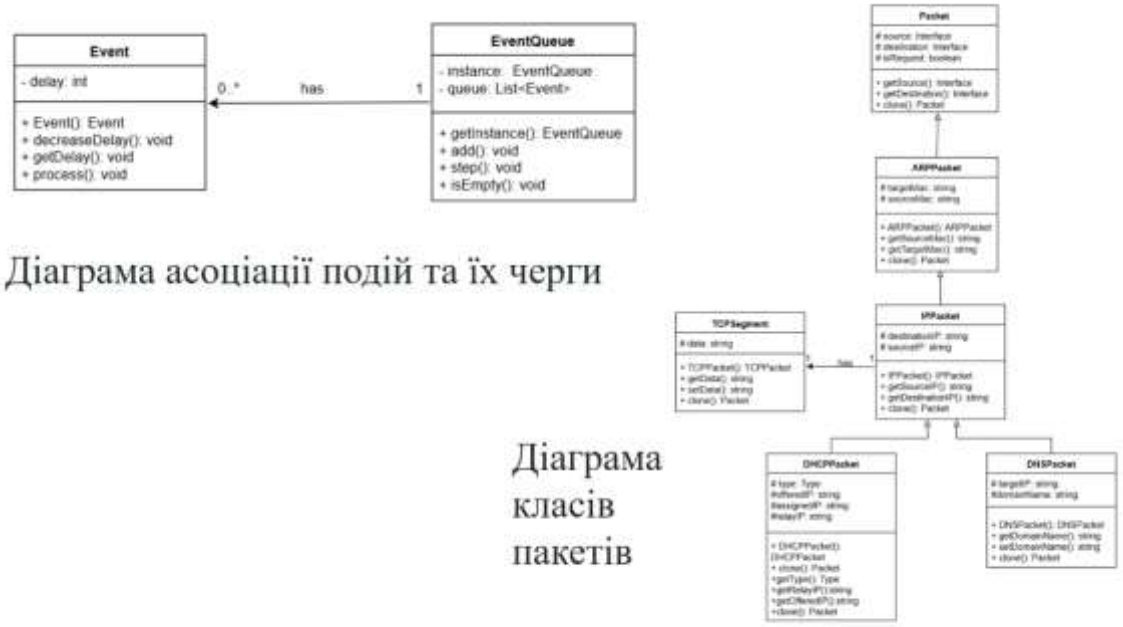


Рисунок Г.11 – Діаграми класів

Графічна схема головного вікна Network Simulator

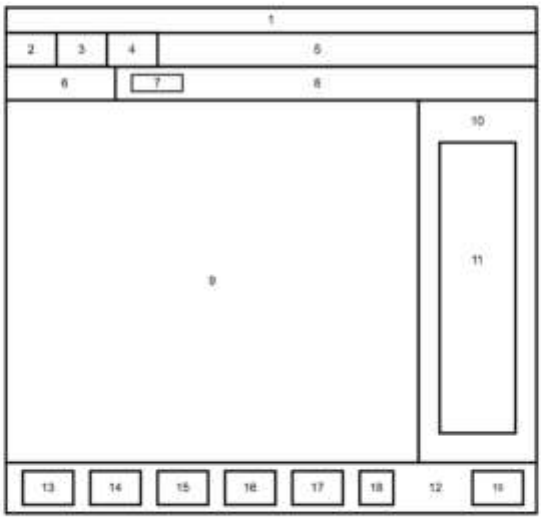


Рисунок Г.12 – Графічна схема

Стартове вікно застосунку

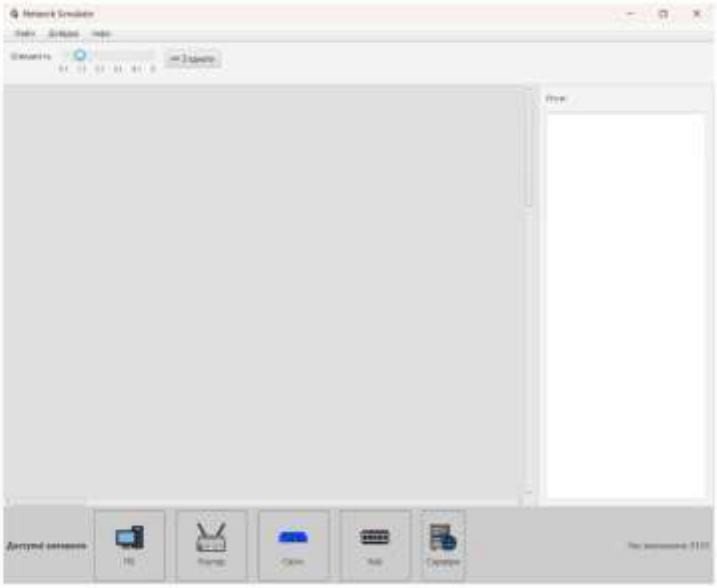


Рисунок Г.13 – Стартове вікно застосунку

Вікна для конфігурації та взаємодії з пристроями

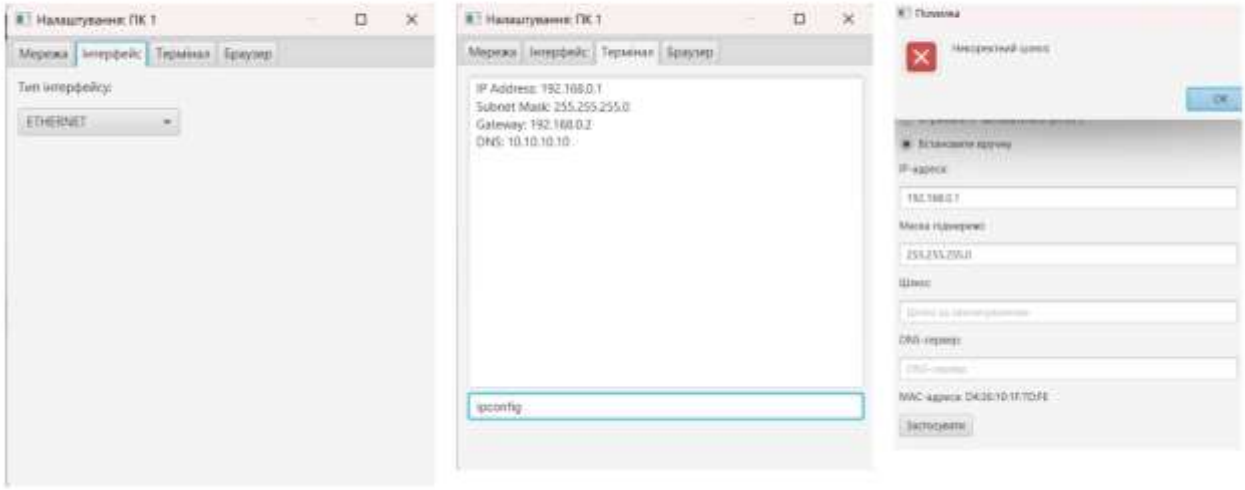


Рисунок Г.14 – Вікна для конфігурації та взаємодії з пристроями

Вікна для конфігурації та взаємодії з пристроями

Рисунок Г.15 – Вікна для конфігурації та взаємодії з пристроями

Чат зі штучним інтелектом

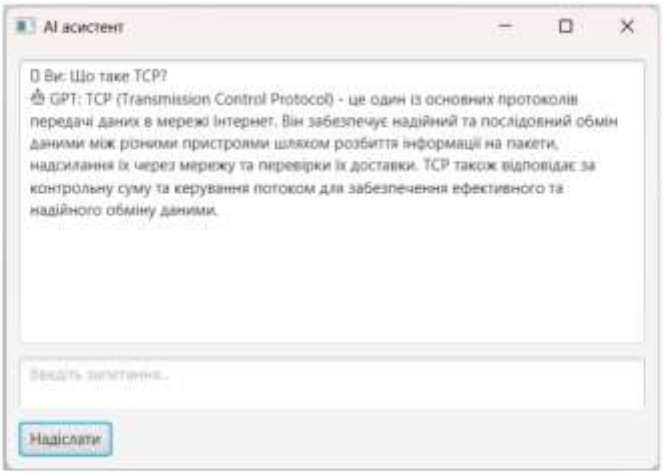


Рисунок Г.16 – Вікно чату зі штучним інтелектом

Калькулятор підмережі

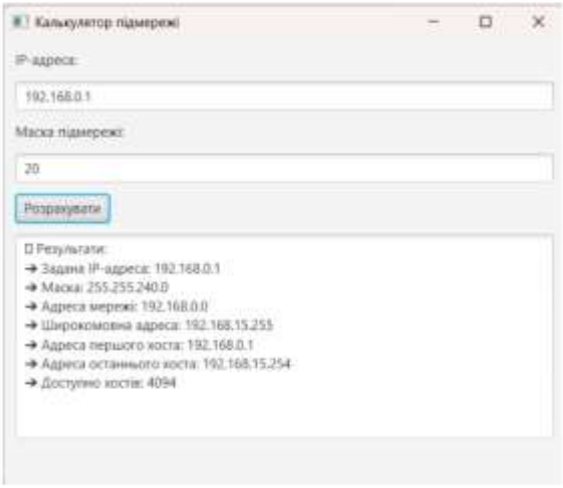


Рисунок Г.17 – Калькулятор підмережі

Довідник

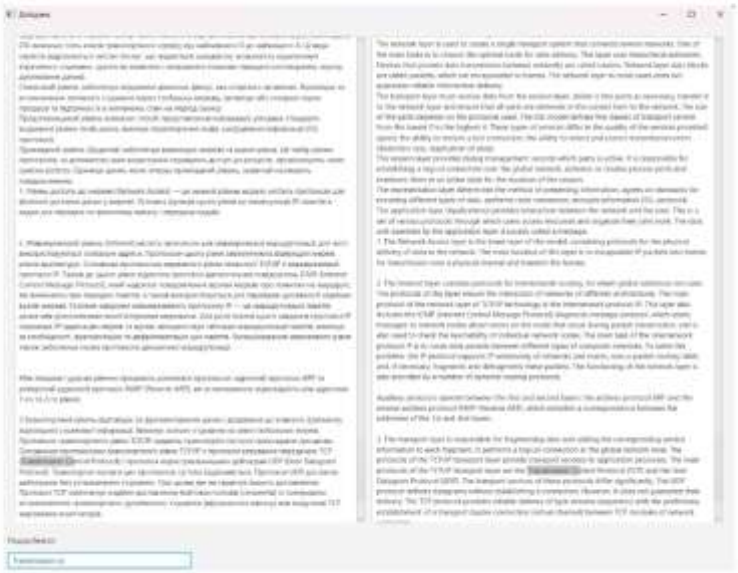
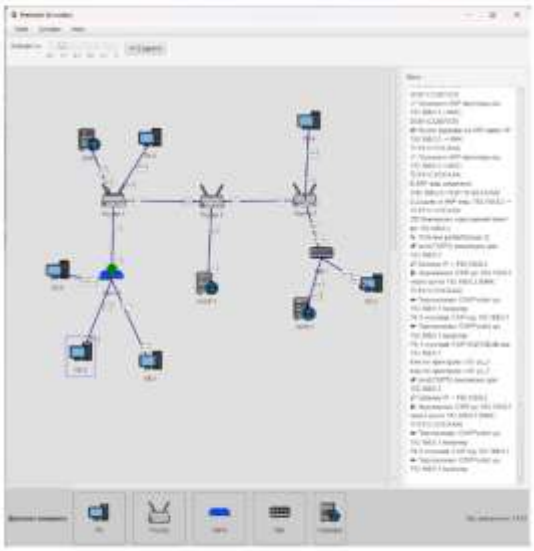


Рисунок Г.18 – Вікно довідника

Тестування



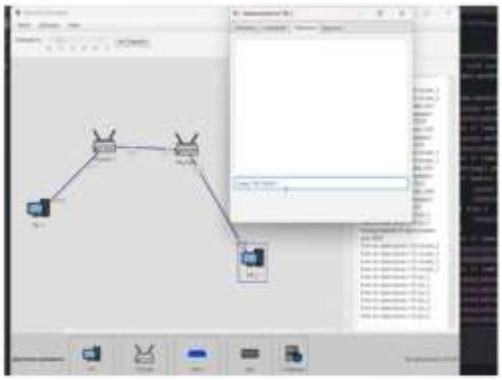


Рисунок Г.19 – Тестування графічного симулятора

Апробація

Основні положення бакалаврської кваліфікаційної роботи доповідалися та обговорювалися на конференціях:

- LIV Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (ВНТУ, 2025);
- I Міжнародній науково-практичній конференції «Innovations in Science: From Theoretical Foundations to Practical Impact» (Антверпен, 2025).

Публікації

Результати роботи опубліковані в двох наукових працях:

- тезах-доповідях на Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії (ВНТУ, 2025);
- тезах-доповідях на I Міжнародній науково-практичній конференції «Innovations in Science: From Theoretical Foundations to Practical Impact» (Антверпен, 2025).

Рисунок Г.20 – Апробація та публікації

Висновки

- проаналізовано засоби та методи для реалізації симулятора мережевих технологій;
- розроблено архітектуру та модель роботи симулятора;
- створено алгоритми взаємодії рівнів моделі OSI та TCP/IP;
- створено графічний інтерфейс застосунку;
- реалізовано модульну структуру симулятора, що забезпечить інтерактивність і наочність навчання;
- проведено тестування і налагодження розробленого програмного продукту.

Рисунок Г.21 – Висновки