Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

зі спеціальності 121 – Інженерія програмного забезпечення

На тему: Розробка пакету TCP/IP утиліт для діагностики мережі

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ–21-1

_____/Б.С. Недзельський/

Керівник кваліфікаційної роботи _____/ Д. В. Швець /

Завідувач кафедри _____/ А. М. Стрюк /

Кривий Ріг

2025

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: бакалавр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри

_____ А. М. Стрюк

«____» _____ 2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІПЗ–21-1 Недзельському Богдану Сергійовичу

1. Тема: «Розробка пакету TCP/IP утиліт для діагностики мережі» затверджена наказом по КНУ № 199с від «10» квітня 2025 р.

2. Термін подання студентом закінченої роботи: «30» травня 2025 р.

3. Вихідні дані по роботі: розроблений пакет прикладних програм має забезпечити реалізацію мережевих утиліт.

4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): провести аналіз існуючих на ринку аналогічних програмних продуктів, обґрунтувати функціонал розроблюваної системи, створити програмне забезпечення, здійснити тестування розробленого додатку.

5. Перелік ілюстративного матеріалу: функціональна схема, блок–схема алгоритму, зображення екранних форм додатку.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Розгляд літературних джерел та пошук інтернет-ресурсів з заданої тематики	03.01.25 – 15.01.25
2	Аналіз існуючих методів вирішення проблеми	16.01.25 – 06.02.25
3	Формулювання актуальності роботи і постановка завдань	07.02.25 – 18.02.25
4	Оформлення матеріалів першого розділу роботи	19.02.25 – 04.03.25
5	Створення функціональної системи та алгоритму додатку	05.03.25 – 15.03.25
6	Оформлення матеріалів другого розділу роботи	16.04.25 – 07.04.25
7	Розробка баз даних, інтерфейсу програмного забезпечення, програмних модулів	08.04.25 – 01.05.25
8	Оформлення додатків	02.05.25 – 07.05.25
9	Тестування розробленої програми	08.05.25 – 15.05.25
10	Оформлення пояснювальної записки	16.05.25 – 29.05.25

Дата видачі завдання: «02» січня 2025 р.

Студент: _____ / Б. С. Недзельський /

Керівник роботи: _____ / Д. В. Швець /

РЕФЕРАТ

МЕРЕЖА, TCP/IP, НАЛАГОДЖЕННЯ, ПІНГ, DNS, WHIOS, УТИЛІТИ, ДОДАТОК.

Пояснювальна записка: 71 с., 21 рис., 1 дод., 15 джерел.

Мета кваліфікаційної роботи: розробка пакету TCP/IP утиліт для діагностики мережі.

Об’єкт проектування: пакет TCP/IP утиліт для діагностики мережі.

У теоретичній частині роботи виконано аналіз існуючих на сьогодні аналогічних програмних рішень на ринку. Зазначені сильні та слабкі сторони існуючих програмних продуктів. Обґрунтовані актуальність роботи, мета та сформульовані завдання для розробки зазначеної системи.

У практичній частині кваліфікаційної роботи реалізовано функціональну схему розроблюваного продукту та алгоритм його роботи. Розроблено інтерфейс програмного продукту, програмну логіку роботи додатку. Проведено тестування розробленого програмного забезпечення.

Розроблений пакет утиліт для діагностики мережі може використовуватися як звичайними користувачами для налаштування домашніх і робочих мереж, а також бути задіяним адміністраторами великих компаній.

ABSTRACT

NETWORK, TCP/IP, CONFIGURATION, PING, DNS, WHIOS, UTILITIES, APPLICATION.

Explanatory note: 71 p., 21 fig., 1 app., 15 references.

The aim of the qualifying work: development of a package of TCP/IP utilities for network diagnostics.

Design object: a package of TCP/IP utilities for network diagnostics.

In the theoretical part of the work, an analysis of today's similar software decisions in the market have executed. The strengths and weaknesses of existing software products have listed. The urgency of work, the purpose, and the objectives for developing the specified system have formulated.

The functional scheme of the developed product and its algorithm has realized in the practical part of the qualification work. The interface of the software product, the program logic of the application have developed. The developed software has tested.

The developed network diagnostic utility package can be used by ordinary users to configure home and work networks, as well as by administrators of large companies.

ЗМІСТ

ВСТУП

На сьогоднішній день надійна робота мережевої інфраструктури є критично важливою умовою функціонування комплексу інформаційних технологій, в першу чергу для забезпечення безперебійного обміну даними, доступу до ресурсів, функціонування хмарних сервісів, корпоративних додатків та систем реального часу.

Зважаючи на зростання обсягів переданих даних, складність архітектури мереж, а також зростаючі вимоги до продуктивності, питання діагностики, моніторингу та аналізу стану комп'ютерних мереж набувають особливої актуальності. Виявлення проблем із підключенням, втратою пакетів, високою затримкою, неправильними налаштуваннями DNS або портів є типовими завданнями, які потребують ефективного інструментарію.

Існуючі утиліти, що працюють у межах TCP/IP-стеку, зокрема Ping, Traceroute, Netstat, Nslookup та інші, широко використовуються системними адміністраторами, інженерами з мережевої безпеки та технічною підтримкою. Однак багато програмних комплексів, що агрегують ці утиліти в єдиному рішенні, мають застарілий інтерфейс, проблеми з сумісністю при використанні в актуальних версіях операційних систем або недостатню гнучкість для автоматизації діагностичних процесів.

У зв'язку з цим актуальною є задача розробки власного пакету TCP/IP утиліт, що забезпечить гнучкий та уніфікований інструментарій для діагностики мережевих з'єднань.

Метою дипломної роботи є розробка програмного пакету утиліт на базі стеку TCP/IP для виконання основних діагностичних операцій у комп'ютерних мережах.

Об'єктом дослідження є процеси діагностики комп'ютерних мереж на основі протоколів TCP/IP.

Предметом дослідження є методи, алгоритми та програмні засоби реалізації мережевої діагностики.

Методами дослідження є аналіз специфікацій протоколів TCP/IP, моделювання мережевих з’єднань, проєктування та програмна реалізація діагностичних інструментів, а також експериментальне тестування.

Дипломна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатку. У роботі розглядаються теоретичні основи функціонування TCP/IP, аналізуються існуючі програмні рішення, описується процес розробки пакету утиліт, а також наводяться результати тестування і приклади використання.

1 РОЗГЛЯД ОСНОВ МЕРЕЖЕВОЇ ВЗАЄМОДІЇ ТА АНАЛІЗ ПОТРЕБ У СТВОРЕННІ ПАКЕТУ TCP/IP УТИЛІТ ДЛЯ ДІАГНОСТИКИ МЕРЕЖІ

1.1 Загальні основи мережевих протоколів TCP/IP

Стек протоколів Transmission Control Protocol / Internet Protocol (TCP/IP) є основою функціонування глобальної мережі Інтернет та більшості сучасних комп'ютерних мереж. Він забезпечує структурований підхід до обміну даними між пристроями, незалежно від особливостей їх апаратного забезпечення або програмної платформи. Стек TCP/IP побудований за принципом модульності, де кожен рівень відповідає за окрему функціональність у процесі передачі інформації.

У загальному вигляді модель TCP/IP складається з чотирьох рівнів: прикладного, транспортного, мережевого та рівня мережевого доступу. При цьому її можна умовно порівняти з еталонною семирівневою моделлю OSI, яка використовується в теорії комп'ютерних мереж як стандарт для класифікації мережевих функцій. Прикладний, представницький і сеансовий рівні моделі OSI відповідають єдиному прикладному рівню в стеку TCP/IP. Транспортний рівень зберігає аналогічну функціональність в обох моделях, тоді як мережевий, канальний та фізичний рівні OSI об'єднуються у стеку TCP/IP в два окремі рівні — мережевий та рівень доступу до мережі.

Прикладний рівень у стеку TCP/IP відповідає за взаємодію користувача з мережею через прикладні сервіси, що реалізують функціональність електронної пошти, передачі файлів, веб-доступу тощо. Саме на цьому рівні функціонують протоколи високого рівня, які забезпечують специфічні види обміну інформацією між додатками.

Транспортний рівень забезпечує логічне з'єднання між вузлами мережі, визначає порядок доставки даних, гарантує їхню цілісність або, навпаки, дозволяє працювати з ненадійними, але швидкими потоками передавання. На

цьому рівні реалізовано два основні підходи: з'єднання з контролем доставки, як у випадку протоколу TCP, та зв'язок без гарантій, як у протоколі UDP, який дозволяє зменшити накладні витрати за рахунок відмови від підтвердження передачі.

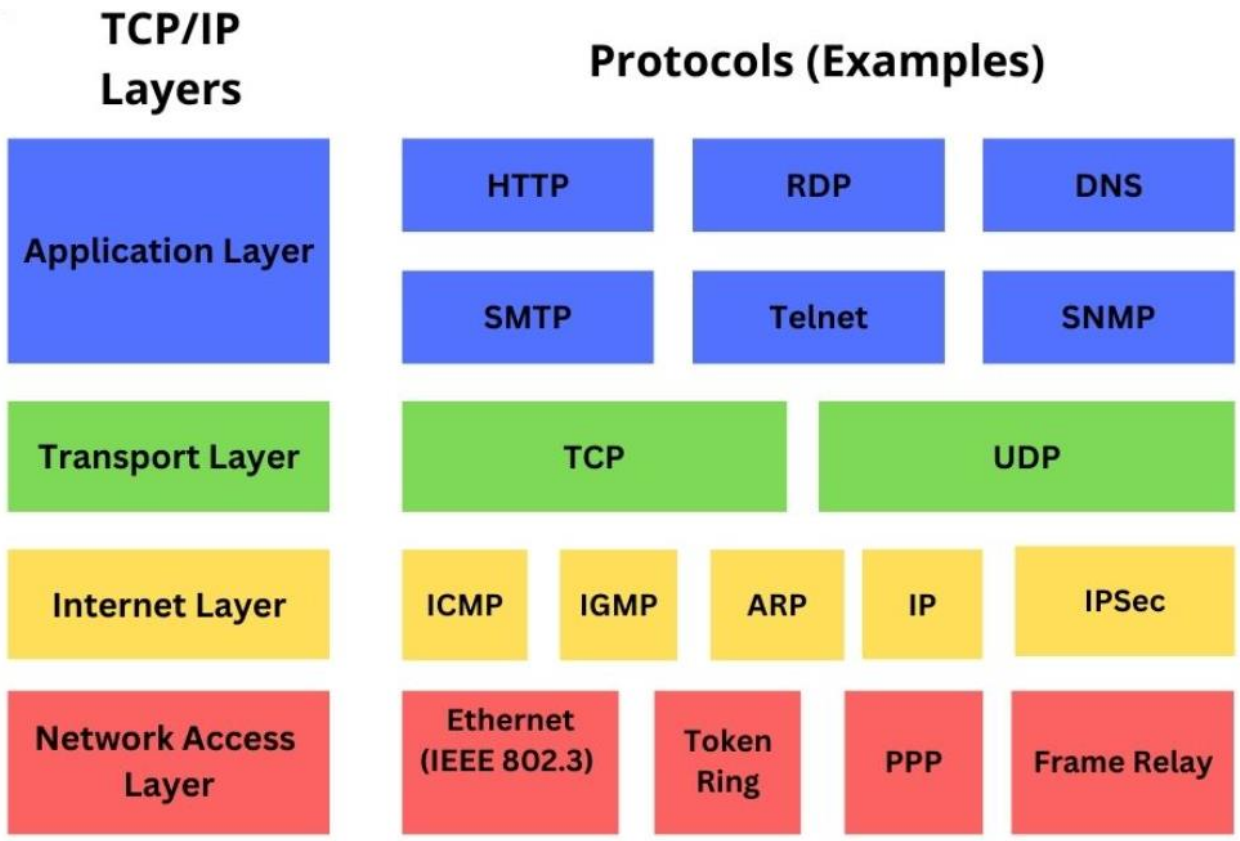

Рисунок 1.1 – Рівні моделі TCP/IP [1]

Мережевий рівень призначений для логічної адресації вузлів і маршрутизації пакетів даних між ними. На цьому рівні реалізовано протокол IP, який визначає схему адресації та механізми передавання даних між підмережами. Допоміжні протоколи, такі як ICMP або ARP, відіграють важливу роль у діагностиці з'єднань і забезпеченні коректної маршрутизації.

Рівень доступу до мережі охоплює як канальні, так і фізичні аспекти передавання даних. Він визначає правила доступу до фізичного середовища, формати кадрів, методи кодування сигналів і взаємодію з мережевими інтерфейсами. Цей рівень безпосередньо залежить від типу середовища

передавання - дротового чи бездротового - і відповідних протоколів, що його обслуговують [2-4].

У сукупності ці чотири рівні створюють модель, яка дозволяє реалізовувати механізми передавання, маршрутизації, захисту та діагностики даних.

Розуміння структури та логіки функціонування TCP/IP є необхідною передумовою для ефективного проєктування мережевих утиліт, які використовуються для аналізу, контролю та діагностики з'єднань у комп'ютерних мережах.

1.2 Модель OSI та її відповідність стеку TCP/IP

Модель OSI, створена Міжнародною організацією зі стандартизації, має переважно концептуальний характер і служить еталоном для опису функціонування відкритих систем взаємодії в телекомунікаційних мережах. Вона розроблялася з метою встановлення універсального підходу до побудови сумісних протоколів, що дозволяють різнорідним інформаційним системам ефективно обмінюватися даними. На відміну від практично орієнтованих стеків, модель OSI не є безпосередньо впроваджуваною в програмному чи апаратному забезпеченні, однак її архітектура широко застосовується як аналітична рамка в проєктуванні і тестуванні мережевих рішень.

Конструкція цієї моделі ґрунтується на принципі розмежування функцій за логічними рівнями, де кожен рівень виконує чітко визначений обсяг завдань і взаємодіє з суміжними рівнями через стандартизовані інтерфейси. Такий підхід дозволяє ізолювати складність, забезпечити гнучкість у розвитку окремих компонентів і спростити верифікацію архітектури мережі. Ключовим у цій моделі є уявлення про ієрархічну структуру, яка дає змогу послідовно описувати обробку даних — від фізичної передачі сигналу до взаємодії прикладних програм.

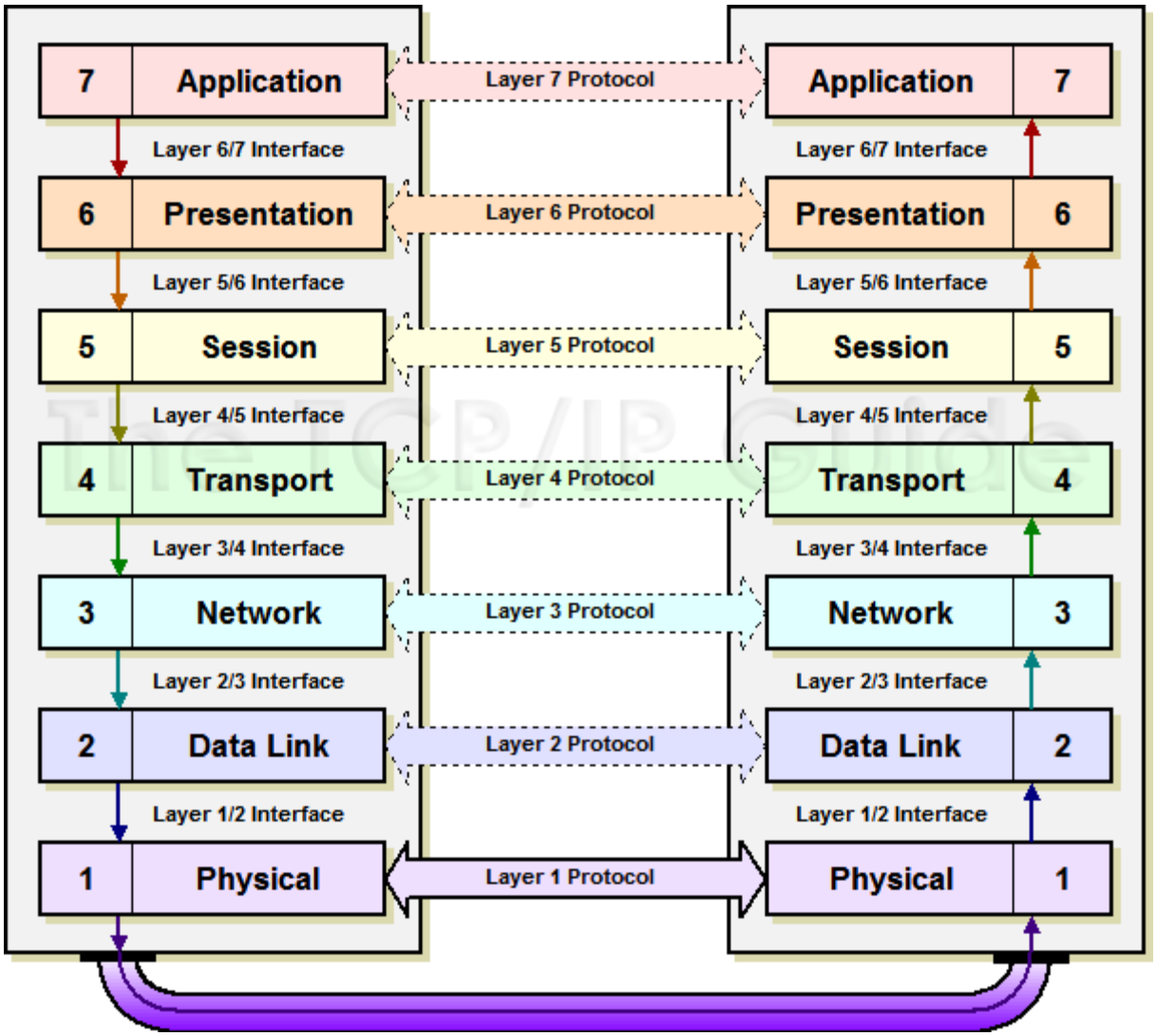

Рисунок 1.2 – Модель OSI [5]

Попри те, що модель OSI не стала домінуючим стандартом у сфері практичного мережевого програмування, вона залишається основою навчання, формалізації технічної документації, а також логічною опорою при створенні утиліт аналізу та діагностики. Саме через неї зручно описувати, на якому рівні відбувається збій або де слід шукати джерело проблеми в роботі мережі. Наприклад, помилки у маршрутизації доцільно відносити до мережевого рівня, тоді як порушення протоколів шифрування — до представницького [6,7].

На тлі концептуальної моделі OSI стек TCP/IP сформувався еволюційно, як відповідь на конкретні потреби військових та наукових мереж. Його

структура визначалася не теоретичною логікою, а вимогами до сумісності, масштабованості та мінімізації складності реалізації. Принципова різниця полягає в тому, що TCP/IP об’єднує деякі рівні OSI і не приділяє окремої уваги таким аспектам, як керування сесіями або форматування даних. Натомість акцент робиться на практичній передачі даних між вузлами в умовах розподіленого середовища з потенційно ненадійними каналами зв’язку.

Оскільки обидві моделі відображають схожі процеси, між ними існує умовна відповідність, яка часто використовується як допоміжний інструмент при розробці і налагодженні мережевого програмного забезпечення. Така відповідність не є жорсткою — вона служить радше для методичного аналізу, ніж для прямого зіставлення структур. У сфері діагностики мережевих збоїв ця відповідність дозволяє точніше локалізувати проблему і вибрати відповідний інструмент дослідження, орієнтуючись як на логіку OSI, так і на механізми TCP/IP.

Таким чином, модель OSI виконує ключову роль у формуванні загального підходу до розуміння архітектури мереж. Вона забезпечує необхідну абстракцію, яка дозволяє незалежно оцінювати якість функціонування окремих компонентів мережевого середовища та служить корисною основою при розробці засобів технічної діагностики, що є безпосередньо пов’язаним із завданнями даної дипломної роботи.

Транспортний рівень у обох моделях виконує однакову функцію — забезпечує передачу даних між хостами, незалежно від того, як саме ці хости з’єднані у фізичному або логічному сенсі. У стеку TCP/IP він реалізує протоколи з різним ступенем надійності, що дозволяє оптимізувати передавання даних під специфічні задачі — від надійної доставки з підтвердженням до швидкої передачі без контролю цілісності.

Мережевий рівень у моделі OSI і стеку TCP/IP також має схожу роль - він відповідає за адресацію та маршрутизацію пакетів даних між мережами. Однак у TCP/IP значна частина функціональності нижчих рівнів OSI, таких як канальний і фізичний, винесена в рівень доступу до мережі, де реалізуються

протоколи взаємодії з конкретним середовищем передавання - кабелем, радіоканалом, оптичним волокном тощо.

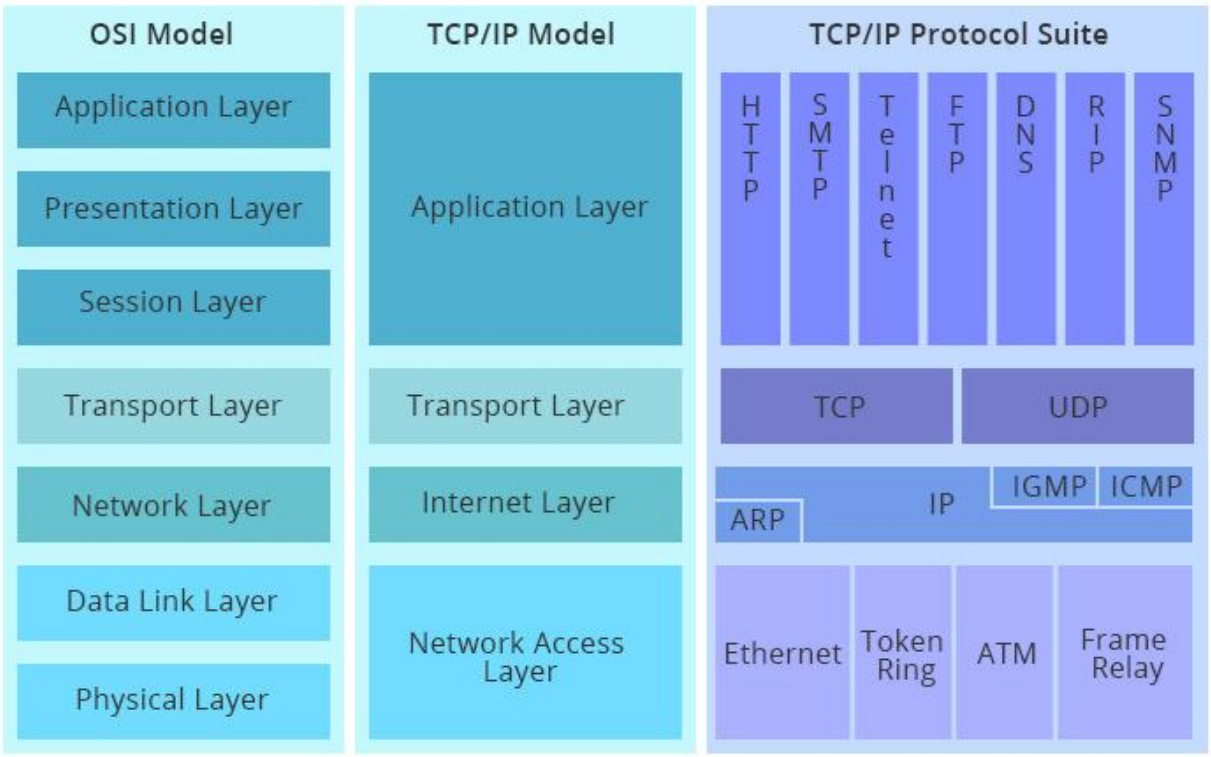

Рисунок 1.3 – Порівняння моделей OSI [8]

Незважаючи на відмінності у кількості рівнів і деталізації функцій, між моделлю OSI та стеком TCP/IP існує досить чітка відповідність. Ця відповідність дозволяє розглядати TCP/IP як приклад реалізації принципів OSI у практичному програмно-апаратному середовищі. Розуміння цієї відповідності є важливим для розробки утиліт, що працюють на межі різних рівнів мережевої взаємодії — зокрема, таких, які діагностують маршрути, тестують доступність хостів, аналізують відкриті порти або перевіряють правильність роботи систем імен.

Таким чином, обидві моделі — і теоретична OSI, і практичний стек TCP/IP — утворюють концептуальну базу, яка дозволяє ефективно описувати, аналізувати та проектувати засоби мережевої діагностики, що є ключовою метою даної дипломної роботи.

1.3 Призначення та класифікація мережевих діагностичних утиліт

Як було зазначено вище, у процесі експлуатації комп'ютерних мереж постійно виникає потреба у перевірці їхньої працездатності, виявленні проблем із передачею даних, оцінці якості з'єднань та контролі доступності окремих вузлів. Для вирішення цих завдань використовуються спеціалізовані програмні засоби, відомі як мережеві діагностичні утиліти. Їх головне призначення полягає у зборі, аналізі та візуалізації технічної інформації про стан мережі, що дозволяє адміністраторам і користувачам ефективно реагувати на відхилення в її роботі.

Діагностика мереж охоплює широкий спектр сценаріїв — від елементарної перевірки доступності окремого хосту до глибокого аналізу маршрутів, стану портів, швидкості передачі, втрат пакетів або часу відгуку. Утиліти, що реалізують такі функції, можуть застосовуватися як у ручному режимі, коли користувач сам ініціює перевірку, так і у складі автоматизованих систем моніторингу, де перевірки виконуються регулярно або за певними умовами.

```
Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Matt>tracert 8.8.8.8

Tracing route to google-public-dns-a.google.com [8.8.8.8]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  192.168.10.254
  2     4 ms     7 ms     1 ms  n4l-akl-internet.mdr-bng1.as45177.net.nz [14.1.43.222]
  3     1 ms     1 ms     1 ms  ae3-1303.mdr-cr1.as45177.net.nz [120.136.0.131]
  4    24 ms    24 ms    25 ms  xe-4-0-1-0.sy3-cr1.as45177.net.au [120.136.0.118]
  5    24 ms    24 ms    24 ms  as15169-ip-119.cust.sy3-cr1.as45177.net.au [120.136.0.119]
  6    25 ms    25 ms    25 ms  216.239.40.233
  7    25 ms    25 ms    25 ms  216.239.40.255
  8    25 ms    25 ms    25 ms  google-public-dns-a.google.com [8.8.8.8]

Trace complete.
```

Рисунок 1.4 – Приклад використання утиліти tracert [9]

Існують засоби, орієнтовані на діагностику логічного з'єднання між пристроями. Вони дозволяють з'ясувати, чи можливо встановити з'єднання на базовому рівні, не вдаючись до аналізу вмісту переданих даних. Інші інструменти фокусуються на відстеженні маршрутів пакетів крізь мережу, що дає змогу виявляти затримки або вузькі місця між окремими ділянками інфраструктури. Окрему групу становлять утиліти, що аналізують активність на рівні портів, дозволяючи перевіряти доступність сервісів, діагностувати помилки взаємодії між клієнтом і сервером, а також виявляти потенційні загрози безпеці.

Також варто відзначити існування універсальних утиліт, які надають комплексну інформацію про поточну мережеву конфігурацію, IP-адресацію, активні інтерфейси, таблиці маршрутизації та інші параметри, що дозволяють формувати загальне уявлення про мережеве середовище конкретного пристрою.

```
┌──(andy㉿kali)-[~]
└─$ whois netscionline.com
   Domain Name: NETSCIONLINE.COM
   Registry Domain ID: 1769657113_DOMAIN_COM-VRSN
   Registrar WHOIS Server: whois.fastdomain.com
   Registrar URL: http://www.fastdomain.com
   Updated Date: 2023-12-14T11:55:51Z
   Creation Date: 2012-12-29T17:24:01Z
   Registry Expiry Date: 2024-12-29T17:24:01Z
   Registrar: FastDomain Inc.
   Registrar IANA ID: 1154
   Registrar Abuse Contact Email:
   Registrar Abuse Contact Phone:
   Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
   Name Server: NS1.DIGITALOCEAN.COM
   Name Server: NS2.DIGITALOCEAN.COM
   DNSSEC: unsigned
   URL of the ICANN Whois Inaccuracy Complaint Form: https://www.icann.org/wicf/
>>> Last update of whois database: 2024-03-25T13:01:09Z <<<

For more information on Whois status codes, please visit https://icann.org/epp
```

Рисунок 1.5 – Приклад використання утиліти whois [10]

Класифікація таких утиліт базується не лише на функціональному призначенні, а й на рівнях мережевої моделі, до яких вони належать. Залежно від реалізованих алгоритмів, діагностичний інструмент може працювати з

фізичним інтерфейсом, з транспортними протоколами, з логічною адресацією або ж з прикладними сервісами, з якими взаємодіє користувач.

У контексті проєктування власного пакету утиліт доцільно орієнтуватися на ті задачі діагностики, які зустрічаються найчастіше у практиці: перевірка зв'язності, контроль відповідей від віддалених вузлів, виявлення затримок, перевірка портів, аналіз маршрутизації та тестування продуктивності каналів. Саме реалізація подібного функціоналу дозволяє створити інструмент, що відповідає потребам як адміністратора мережі, так і звичайного користувача, який прагне зрозуміти, чому Інтернет-з'єднання працює нестабільно чи не працює взагалі.

1.4 Аналіз існуючих на програмних засобів для діагностики стану мережевого з'єднання

1.4.1 Advanced IP Scanner

Advanced IP Scanner [11] — це утиліта для операційної системи Windows, яка дозволяє виконувати швидке сканування локальної мережі з метою виявлення активних пристроїв. Вона орієнтована на адміністраторів мереж та IT-фахівців, яким потрібен інструмент для контролю мережевого середовища, перевірки підключень і керування вузлами.

Після запуску програма аналізує вказаний діапазон IP-адрес і визначає пристрої, що знаходяться в мережі та відповідають на запити. Користувач отримує інформацію про IP-адреси, MAC-адреси, імена комп'ютерів і наявність спільних ресурсів. Програма дозволяє не лише переглядати дані, а й виконувати дії, наприклад підключення до віддаленого робочого столу через RDP, доступ до спільних папок або керування комп'ютером через Radmin. Крім того, реалізована підтримка функції Wake-on-LAN, що дозволяє вмикати пристрої в мережі дистанційно.

Програма вирізняється зручністю інтерфейсу та невимогливістю до ресурсів. Її не обов'язково встановлювати, оскільки доступна портативна версія. Результати сканування можна зберігати у файли для подальшого

аналізу. Це робить Advanced IP Scanner інструментом як для щоденного використання, так і для періодичного аудиту мережі.

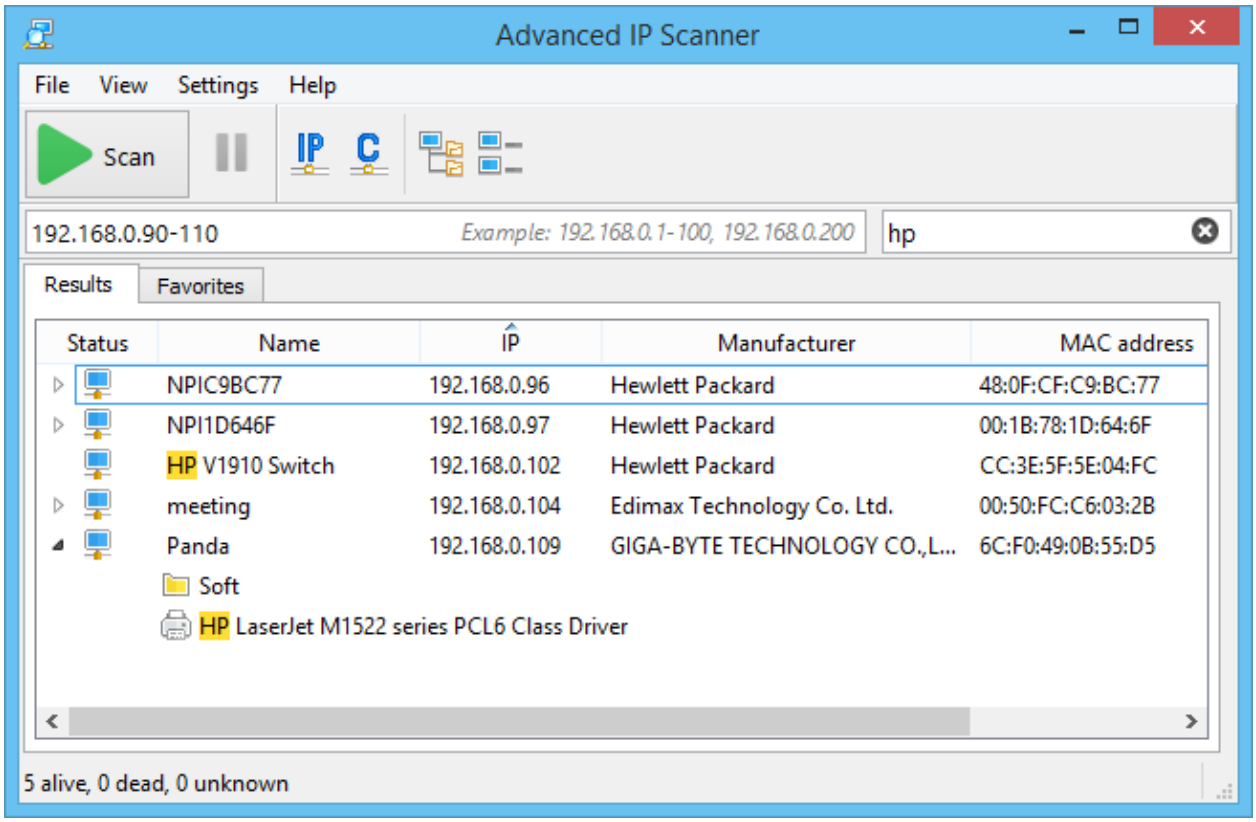

Рисунок 1.6 – Advanced IP Scanner

Загалом, Advanced IP Scanner є ефективним засобом для візуального моніторингу, аналізу активності в мережі та проведення початкової діагностики без необхідності наявності глибоких технічних знань.

Advanced IP Scanner, незважаючи на свою простоту, впевнено зайняв свою нішу серед утиліт для швидкої діагностики локальної мережі. Його популярність пояснюється тим, що програма добре виконує саме ті базові завдання, які найчастіше потребують вирішення в офісних або домашніх умовах.

В той же час необхідно зазначити, що вбудований набір мережевих інструментів є достатньо невеликим.

Водночас можна дійти висновків, що функціональність програми зумисно обмежена — вона не надає глибоких інструментів аналізу або засобів

виявлення складних мережевих проблем. Її логіка побудована навколо максимальної доступності для користувачів без спеціальної технічної підготовки, що, з одного боку, робить інтерфейс зручним, але з іншого - знижує інформативність результатів для досвідчених спеціалістів.

Також програма орієнтована насамперед на роботу в локальному сегменті мережі й не підходить для задач, пов’язаних з аналізом віддалених хостів або складних багаторівневих топологій. Це обмеження робить її менш придатною для мережевих інженерів, які працюють у великих інфраструктурах.

У підсумку, Advanced IP Scanner є зручним інструментом для базового обстеження локальної мережі, проте його варто сприймати саме як початковий рівень діагностики. Для глибшого аналізу або розширеного функціоналу доцільно комбінувати його з іншими спеціалізованими програмами.

1.4.2 Essential NetTools

Essential NetTools [12] - це програмний комплекс для операційної системи Windows, призначений для діагностики мережі, моніторингу з'єднань і аналізу безпеки. Він об'єднує в собі набір інструментів, які дозволяють користувачам досліджувати мережеву активність, виявляти потенційні загрози та оптимізувати роботу мережевих служб.

Однією з функцій Essential NetTools є можливість відображення активних мережевих з'єднань, включаючи інформацію про відкриті порти, IP-адреси та стан мережевих з'єднань. Це дозволяє користувачам відстежувати вхідні та вихідні підключення, а також ідентифікувати програми, які їх ініціюють.

Програма також надає інструменти для сканування мережі, що дозволяє виявляти активні пристрої в заданому діапазоні IP-адрес, аналізувати відкриті порти та перевіряти доступність хостів. Це корисно для виявлення несанкціонованих пристроїв або служб, що працюють у мережі.

Крім того, Essential NetTools включає функції для перевірки електронних адрес на існування, аналізу DNS-записів, моніторингу спільних ресурсів та виявлення потенційних вразливостей у службах NetBIOS. Це дозволяє користувачам забезпечити безпеку мережі та запобігти можливим атакам.

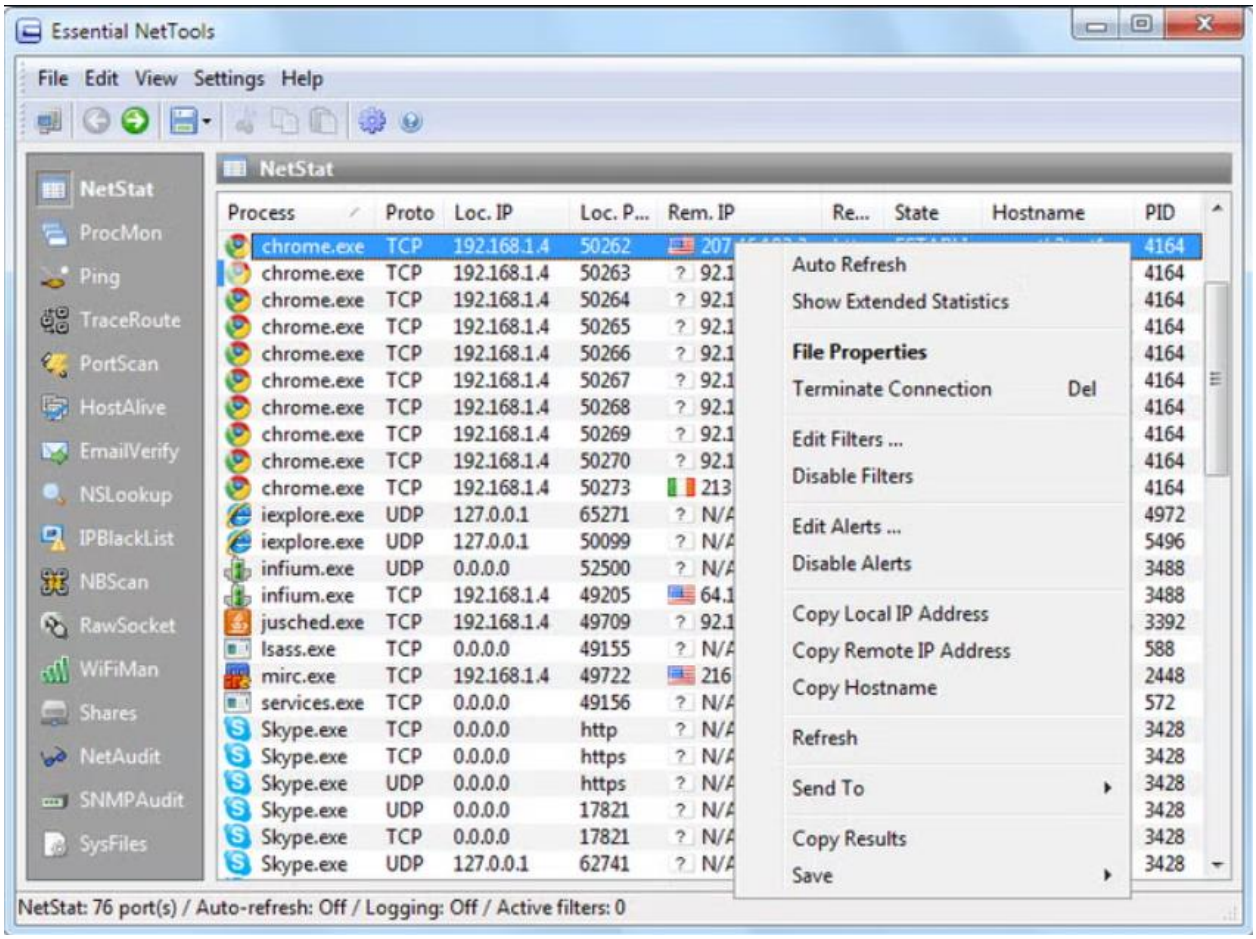

Рисунок 1.7 – Essential NetTools

Інтерфейс програми інтуїтивно зрозумілий, що робить її доступною як для досвідчених адміністраторів, так і для звичайних користувачів.

Хоча Essential NetTools і був популярним серед фахівців завдяки своїй багатофункціональності, його актуальність у сучасних умовах викликає сумніви. Одним із головних факторів, що свідчить про занепад розвитку програми - це відсутність оновлень після 2021 року. Такий стан речей вказує

на ймовірне припинення підтримки з боку розробника, що є критичним фактором.

Також інтерфейс Essential NetTools виглядає не лише морально застарілим, а й менш зручним у порівнянні з сучасними рішеннями, що активно використовують адаптивний дизайн та інтерактивні елементи управління. Це ускладнює роботу для нових користувачів з ним.

Таким чином, попри історичну цінність та значну кількість вбудованих інструментів, Essential NetTools дедалі більше сприймається як приклад програмного забезпечення минулого покоління.

1.4.3 Axence netTools

Axence netTools [13] - це широкофункціональний програмний комплекс для діагностики мереж, який орієнтований на системних адміністраторів, мережевих аналітиків і технічних фахівців. Програма об’єднує в одному інтерфейсі широкий набір інструментів, що дозволяє проводити повноцінний аналіз локальної мережі та окремих хостів, діагностувати проблеми, перевіряти конфігурацію та продуктивність мережевого обладнання.

Інтерфейс програми реалізований у вигляді єдиного централізованого середовища, де користувач може отримати доступ до основних функцій — від сканування мережі та портів до моніторингу трафіку, відстеження відкритих з’єднань і перевірки відповідності служб. Axence netTools дозволяє переглядати IP-адреси, DNS-сервери, відповідь хостів, географічне розташування вузлів, а також виявляти аномальні або підозрілі підключення.

Серед найпомітніших можливостей — система NetWatch, яка забезпечує моніторинг доступності кількох хостів одночасно, з можливістю автоматичного повідомлення у випадку втрати зв’язку чи підвищеного часу відповіді. Програма також зберігає історію даних, що дає змогу аналізувати поведінку мережі в динаміці, експортувати звіти у зручних форматах (XML, HTML, TXT) та зберігати шаблони для повторного використання.

Вбудований SNMP-браузер дає змогу отримувати дані з мережевих пристроїв через стандартні протоколи, а функція TCP/IP Workshop дозволяє досліджувати нестандартні сервіси або низькорівневі з'єднання. Крім цього, користувач отримує доступ до віддаленого перегляду системних процесів, реєстру Windows, журналів подій, а також до інформації про апаратне забезпечення.

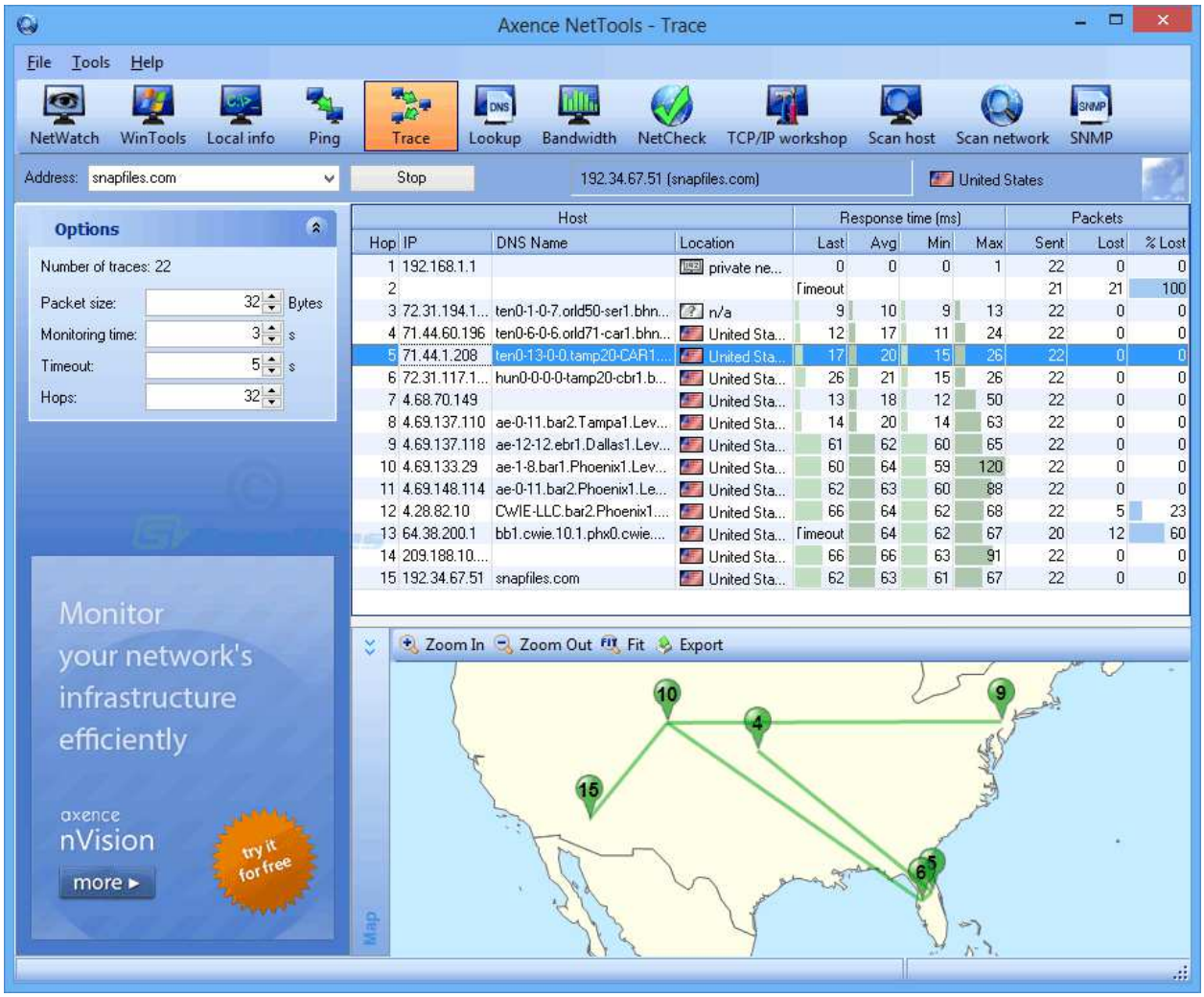

Рисунок 1.8 – Axence netTools [14]

Особливу увагу Axence netTools приділяє перевірці безпеки з'єднань: інструмент Traceroute доповнено геолокаційною візуалізацією, що дозволяє виявляти підозрілі маршрути пакетів або з'єднання з країнами, які можуть викликати занепокоєння. Програма також підтримує перевірку якості

мережевих кабелів (NetCheck), тестування пропускної здатності (Bandwidth) та перегляд актуальних DNS і WHOIS-записів (Lookup).

У підсумку, Axence netTools є комплексним рішенням для глибокого аналізу мережі, яке охоплює як базові задачі на кшталт ping і traceroute, так і розширені можливості моніторингу служб, перевірки апаратних характеристик та аналізу мережевої безпеки.

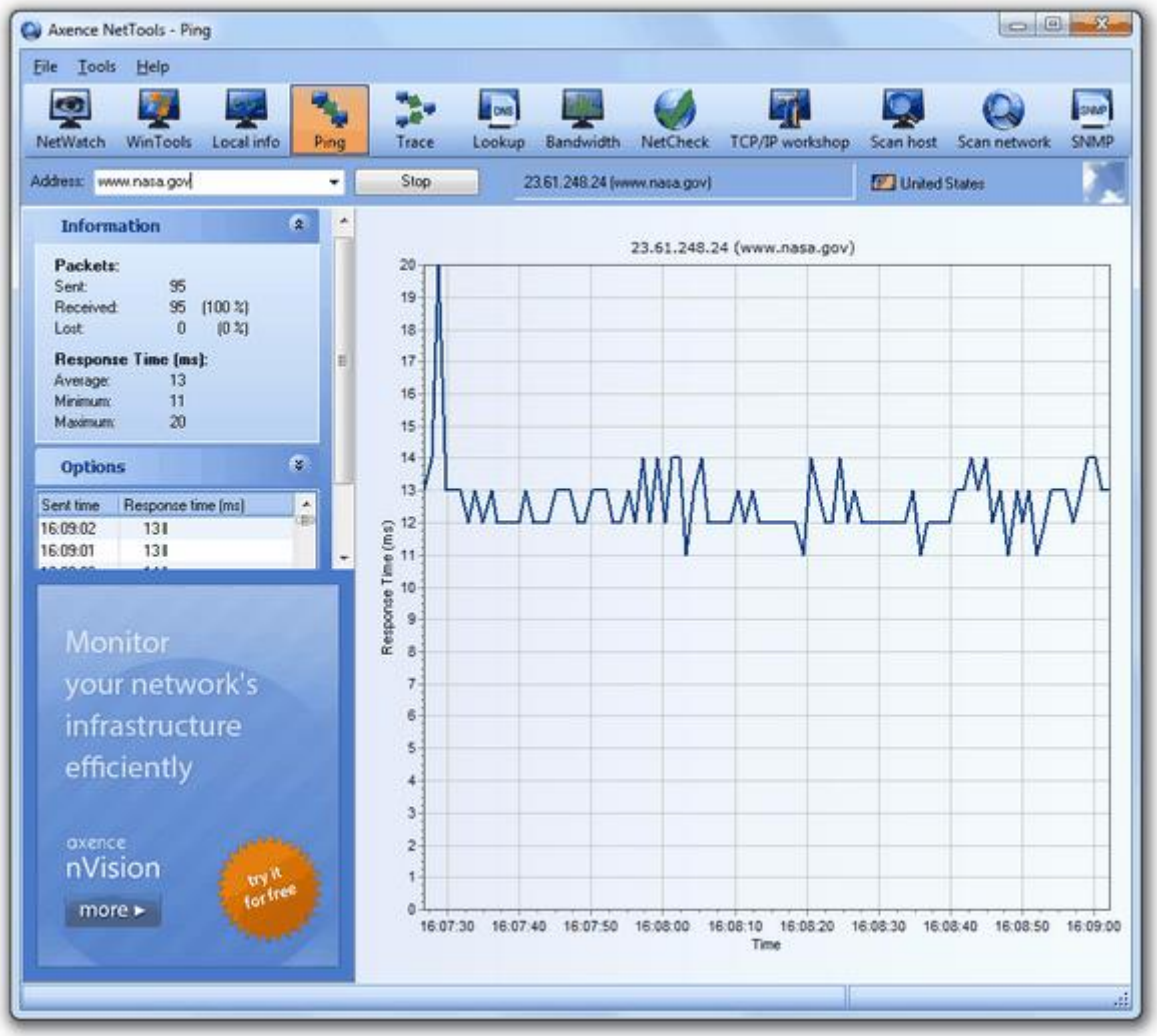

Рисунок 1.9 – Axence netTools [14]

Але хоча Axence netTools вирізняється широкими функціональними можливостями й може слугувати потужним інструментом для діагностики мережі, він має низку недоліків, які обмежують його використання в певних сценаріях.

Насамперед, програма доступна лише для Windows, що робить її непридатною для системних адміністраторів, які працюють у середовищах Linux, macOS або мультиплатформних інфраструктурах. Відсутність кросплатформеності знижує її універсальність у сучасному корпоративному середовищі, де часто використовуються змішані системи.

Інтерфейс, хоча й функціональний, виглядає застарілим у плані дизайну, особливо в порівнянні з сучасними утилітами, які надають адаптивний інтерфейс, інтерактивні графіки та темну тему. Це може створювати враження, що продукт не розвивається досить активно або не відповідає сучасним стандартам UX/UI.

Ще одним недоліком є відсутність централізованої системи логування або звітності, орієнтованої на довготривале спостереження чи аудиторське зберігання даних. Хоча результати можна експортувати вручну, автоматизовані процеси збору та збереження даних реалізовані слабо, що ускладнює роботу в складних мережах або великих організаціях.

Деякі функції, пов’язані з моніторингом, наприклад перевірка стану служб або відстеження втрат пакетів, працюють нестабільно на системах з активними антивірусами або захисниками Windows, які блокують низькорівневі запити. Це може призвести до хибних спрацювань або неповних результатів.

Також варто зазначити, що програма не має вбудованої системи оновлення через інтерфейс — користувачу доводиться самостійно завантажувати нову версію з сайту, що не надто зручно в умовах інтенсивного використання або великої кількості робочих станцій.

Таким чином, попри багатий функціонал, Axence netTools поступається сучасним аналогам у питаннях інтерфейсу, сумісності, автоматизації та гнучкості розгортання, що варто враховувати при виборі інструментів для професійної роботи з мережами.

1.4.4 IP-Tools

IP-Tools [15] -— це універсальний набір діагностичних утиліт TCP/IP, який об’єднано в єдиному програмному середовищі для зручної роботи під керуванням Windows. Програма підтримує сучасні й застарілі версії операційної системи — від Windows XP до Windows 11, а також серверні редакції до Windows Server 2022 включно. Завдяки широкому спектру вбудованих інструментів IP-Tools орієнтована на системних адміністраторів, мережевих інженерів та всіх, хто працює з IP-мережами.

Основна цінність програми полягає в тому, що вона надає понад двадцять утиліт в одному інтерфейсі. Це дозволяє проводити аналіз мережевої інфраструктури на кількох рівнях — від базових перевірок доступності хостів до моніторингу мережевого трафіку, збору системної інформації, аналізу TCP/UDP-з’єднань і роботи з DNS, SNMP, HTTP і Telnet.

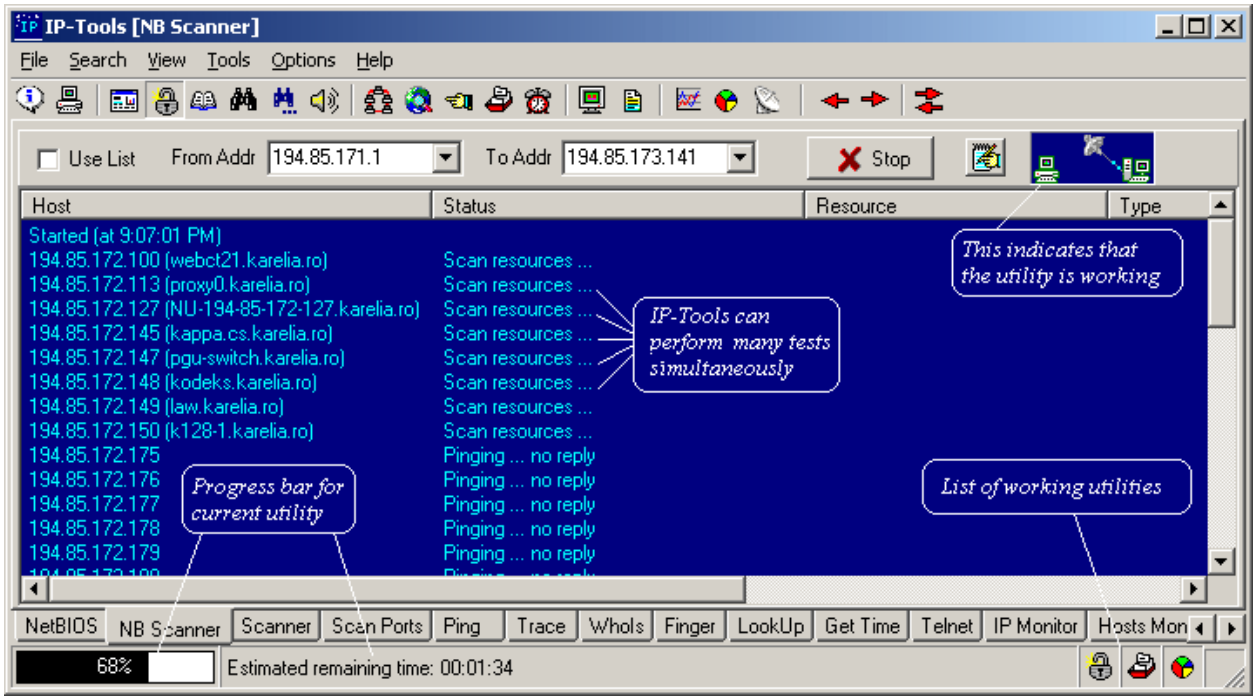

Рисунок 1.10 – IP-Tools

Програма забезпечує сканування портів і сервісів, трасування маршрутів, перегляд відкритих з’єднань, моніторинг хостів, роботу з Whois і

DNS-запитами, а також дозволяє здійснювати перевірку як окремих адрес, так і цілих діапазонів. IP-Tools також підтримує збереження результатів у текстовому або HTML-форматі, що полегшує створення звітів або документування мережевої активності.

Окрім основних можливостей, серед особливостей IP-Tools можна відзначити підтримку одночасного виконання кількох утиліт, гнучку систему налаштувань, простий інтерфейс, а також постійну підтримку усіх майбутніх версій програми після реєстрації. Це робить IP-Tools доволі привабливим вибором для фахівців, які цінують стабільність, універсальність та інтеграцію великої кількості функцій в одному рішенні.

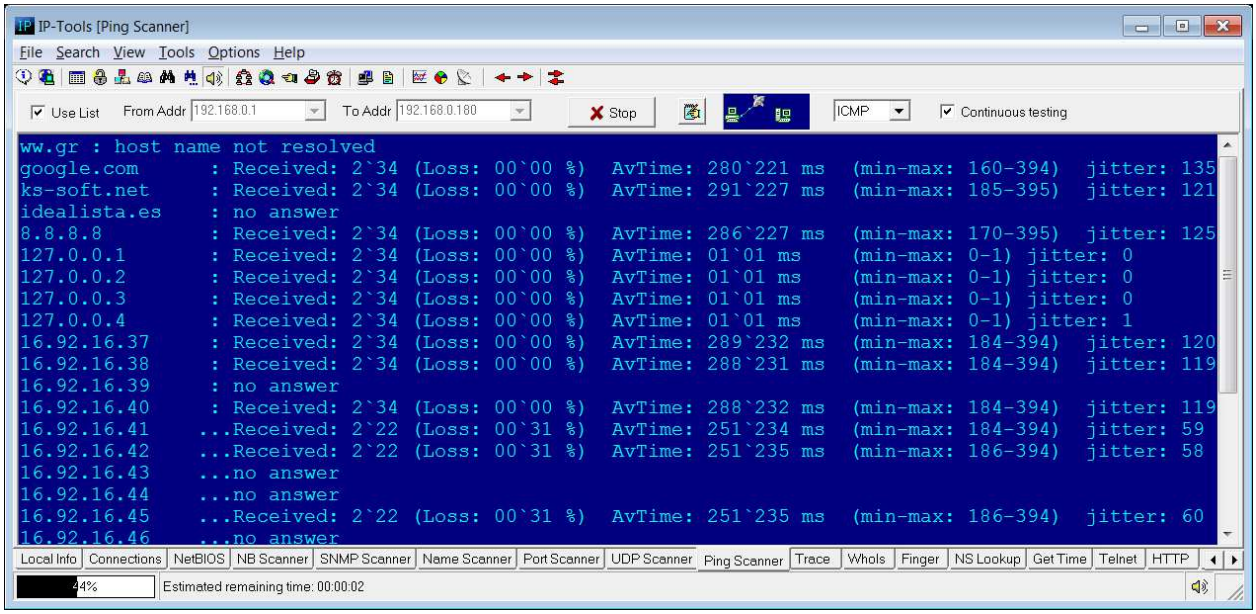

Рисунок 1.11 – IP-Tools

Попри широку функціональність і довгу історію використання, IP-Tools має низку суттєвих недоліків, які обмежують його актуальність у сучасних умовах.

По-перше, програма не демонструє активного розвитку, а інтерфейс залишився практично незмінним протягом багатьох років. Це не лише візуально застаріло, а й функціонально — відсутність сучасних інтерактивних графіків, адаптивного дизайну або підтримки тем оформлення ускладнює

роботу з великим обсягом інформації. У порівнянні з новітніми інструментами, IP-Tools виглядає архаїчно.

По-друге, незважаючи на кількість вбудованих утиліт, їх реалізація досить базова: багато функцій мають обмежену аналітику, мінімум налаштувань та не забезпечують глибокої діагностики або зручної візуалізації результатів. Відсутні, наприклад, інструменти для аналізу мережевої безпеки, сучасного трафіку або взаємодії з хмарними сервісами, які вже стали стандартом у багатьох професійних продуктах.

Також слід зазначити відсутність кросплатформеності — IP-Tools працює виключно під Windows. Це значно звужує сферу його використання, особливо в умовах, коли все більше організацій використовують гібридні або Linux-орієнтовані інфраструктури.

Крім того, деякі інструменти працюють нестабільно в сучасних системах із підвищеним рівнем захисту, таким як Windows із активованим брандмауером, антивірусами або системами контролю облікових записів (UAC). Це може призводити до неповних результатів або необхідності ручної зміни параметрів безпеки.

Нарешті, IP-Tools не підтримує централізоване зберігання історії або логування подій. Уся аналітика зберігається локально й переважно у вигляді текстових файлів, що не підходить для великих або розподілених мереж, де потрібен постійний моніторинг, журналювання і централізований доступ до звітності.

У результаті, хоча IP-Tools усе ще може бути корисним у простих сценаріях або як навчальний інструмент, він суттєво поступається більш сучасним рішенням у плані гнучкості, зручності та глибини аналізу.

1.5 Підбиття висновків по здійсненому аналізу програм для тестування мережі

На основі аналізу наявних програмних рішень для діагностики комп’ютерних мереж можна зробити висновок, що більшість з них мають

спільні недоліки, які обмежують їхню ефективність. Хоча такі утиліти пропонують широкий набір функцій, у багатьох випадках спостерігається відсутність активного розвитку, застарілий інтерфейс і низька адаптивність до нових технологічних вимог.

Основні недоліки полягають у відсутності підтримки кросплатформенності, обмеженій інтеграції з сучасними інструментами адміністрування, неможливості централізованого збору та аналізу даних, а також відсутності зручних механізмів для автоматизації рутинних завдань. Більшість рішень орієнтовані виключно на ручну взаємодію і не враховують потреби командної роботи. Також значна частина існуючих програм не підтримується розробниками або оновлюється вкрай рідко, що призводить до втрати актуальності в умовах змінної мережевої інфраструктури.

У зв’язку з цим виникає потреба у створенні нового, простого та інтуїтивно зрозумілого пакету прикладних програм, який би виконував базові функції діагностики мережі без перевантаження інтерфейсу або складної конфігурації. Такий пакет має зосереджуватись на ключових інструментах - перевірці доступності хостів, трасуванні маршрутів, скануванні портів, перегляді з’єднань, аналізі DNS — і реалізовуватись із урахуванням сучасних вимог до швидкості роботи та безпеки.

Простота, легкість у розгортанні, мінімум залежностей - мають стати базовими характеристиками такого програмного продукту. Реалізація подібного інструменту дозволить ефективно закрити потреби як фахівців-початківців, так і досвідчених мережевих адміністраторів, яким потрібне легке та надійне рішення для щоденних задач.

2 ПРОЕКТУВАННЯ ПАКЕТУ TCP/IP УТИЛІТ ДЛЯ ДІАГНОСТИКИ МЕРЕЖІ

2.1 Аналіз вимог до об'єкта проєктування

Аналіз вимог до об'єкта проєктування передбачає визначення функціонального призначення, очікуваних властивостей та умов використання програмного забезпечення, що створюється в ході кваліфікаційної роботи. Основною метою розробки є створення ефективного і в той же час ергономічного програмного пакету, що забезпечить базові засоби перевірки, тестування та діагностики стану комп'ютерних мереж за допомогою використання протоколів TCP/IP. Програмне забезпечення має працювати в умовах операційної системи Windows, не потребувати встановлення сторонніх бібліотек, забезпечувати стабільну роботу при мережевих помилках і бути орієнтованим на використання як адміністраторами мереж, так і користувачами з базовими знаннями мережевих технологій.

Розглянемо функціональні вимоги до проектованого пакету TCP/IP утиліт для діагностики мережі.

До складу пакету мають входити наступні утиліти:

а) Ping;

б) Traceroute;

в) Port Scanner;

г) DNS Lookup;

д) Whois.

Утиліта Ping має забезпечувати перевірку доступності віддалених вузлів і вимірювання затримки проходження пакетів. Traceroute має визначати маршрут пакетів до цільового хоста, що дає змогу виявити проблемні ділянки маршруту. Port Scanner повинен здійснювати перевірку TCP-портів віддалених хостів на відкритість і визначати стандартні сервіси, що їх використовують.

DNS Lookup має виконувати перетворення доменних імен в IP-адреси. Утиліта Whois повинна забезпечити доступ до реєстраційної інформації про домени або IP-адреси за допомогою протоколу WHOIS.

Що стосується нефункціональних вимог, то усі утиліти повинні мати спільну структуру інтерфейсу, реалізовувати зручне виведення результатів і обробку типових помилок (недоступність мережі, неправильні адреси тощо).

Програмне забезпечення, що реалізує набір TCP/IP утиліт для діагностики мережі, повинно бути написане з використанням мови програмування C# та із застосуванням стандартних бібліотек .NET, що забезпечить швидкість розробки і простоту збірки. Пакет має бути працювати у останніх версіях операційних систем Windows (10, 11). Виведення результатів має бути структурованим і зручним. Програма повинна демонструвати стабільну роботу при відсутності підключення до мережі або за умов обмеженого доступу до ресурсів. Код має бути документованим і структурованим відповідно до принципів підтримуваності й повторного використання. Пакет не повинен потребувати встановлення стороннього програмного забезпечення чи драйверів, за винятком середовища виконання .NET.

У результаті виконання дипломного проєкту очікується отримання функціонального і практичного засобу для діагностики комп’ютерних мереж, придатного для використання в умовах підприємств або при індивідуальній роботі з мережею.

2.2 Діаграми потоків даних пакету TCP/IP утиліт для діагностики мережі

На рисунку 2.1 наведено діаграму потоків даних нульового рівня пакету TCP/IP утиліт для діагностики мережі, а на рисунку 2.2 – відповідно, діаграму потоків даних першого рівня.

Наведемо детальний опис діаграми потоків даних першого рівня пакету утіліт, що розробляється.

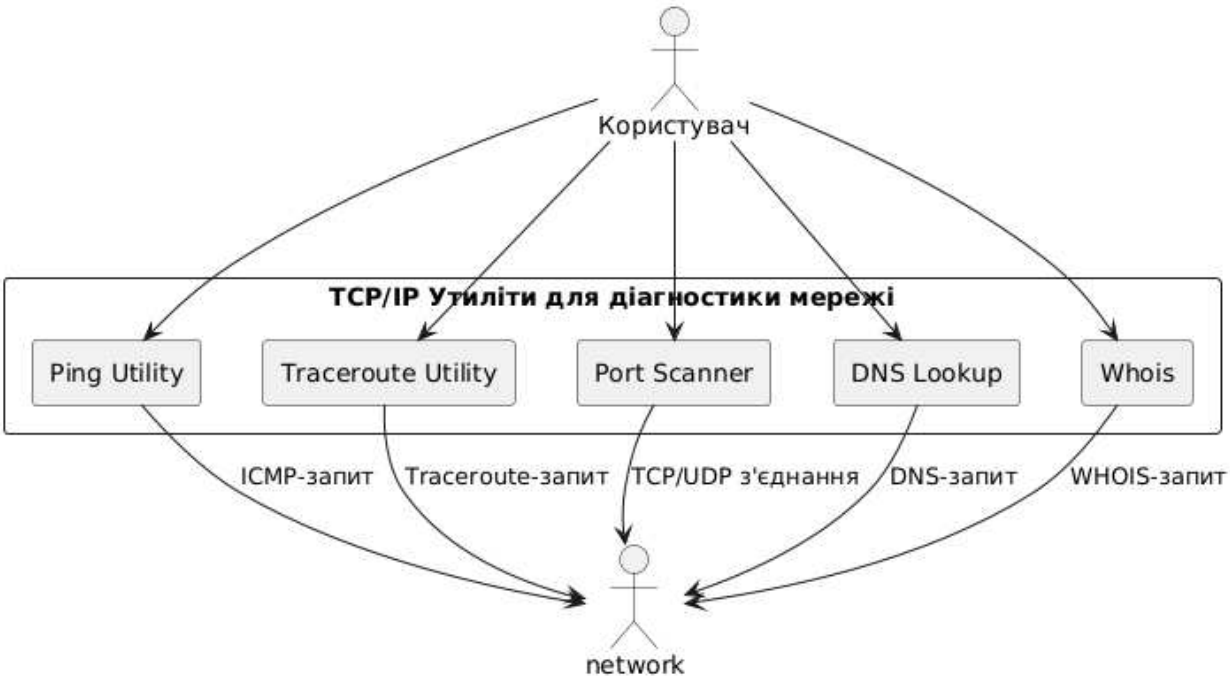

Рисунок 2.1 – Діаграма потоків даних нульового рівня

Програмний пакет "TCP/IP tools" є основним процесом в цій схемі. Користувачем пакету виступає адміністратор або досвідчений користувач, що здійснює взаємодію з п’ятьма основними утилітами пакету, кожна з яких виконує певну функцію діагностики мережі.

Ping Utility отримує від користувача три параметри: IP-адресу, TTL (час життя пакету) та розмір пакету. Ця утиліта надсилає ICMP-запити до мережі і повертає користувачу результати у вигляді часу відгуку, TTL та статусу доступності хоста. Результати дозволяють користувачеві оцінити стан зв'язку з віддаленим хостом.

Traceroute Utility отримує від адміністратора IP-адресу або доменне ім’я. Вона надсилає спеціальні пакети до мережі для визначення маршруту до цільового вузла. У процесі виконання утиліта виявляє проміжні хопи (маршрутизатори) та час проходження пакету, що дозволяє виявити можливі проблеми на конкретних ділянках маршруту. Результати повертаються у вигляді списку хопів та часу проходження кожного етапу.

Port Scanner приймає IP-адресу та діапазон портів. Утиліта здійснює спроби встановити TCP-з’єднання з віддаленим хостом через мережу та

перевіряє відкритість портів. Вона визначає відкриті порти та пов'язані з ними сервіси, а також передає результати у вигляді списку відкритих портів і типів сервісів, що працюють на цих портах.

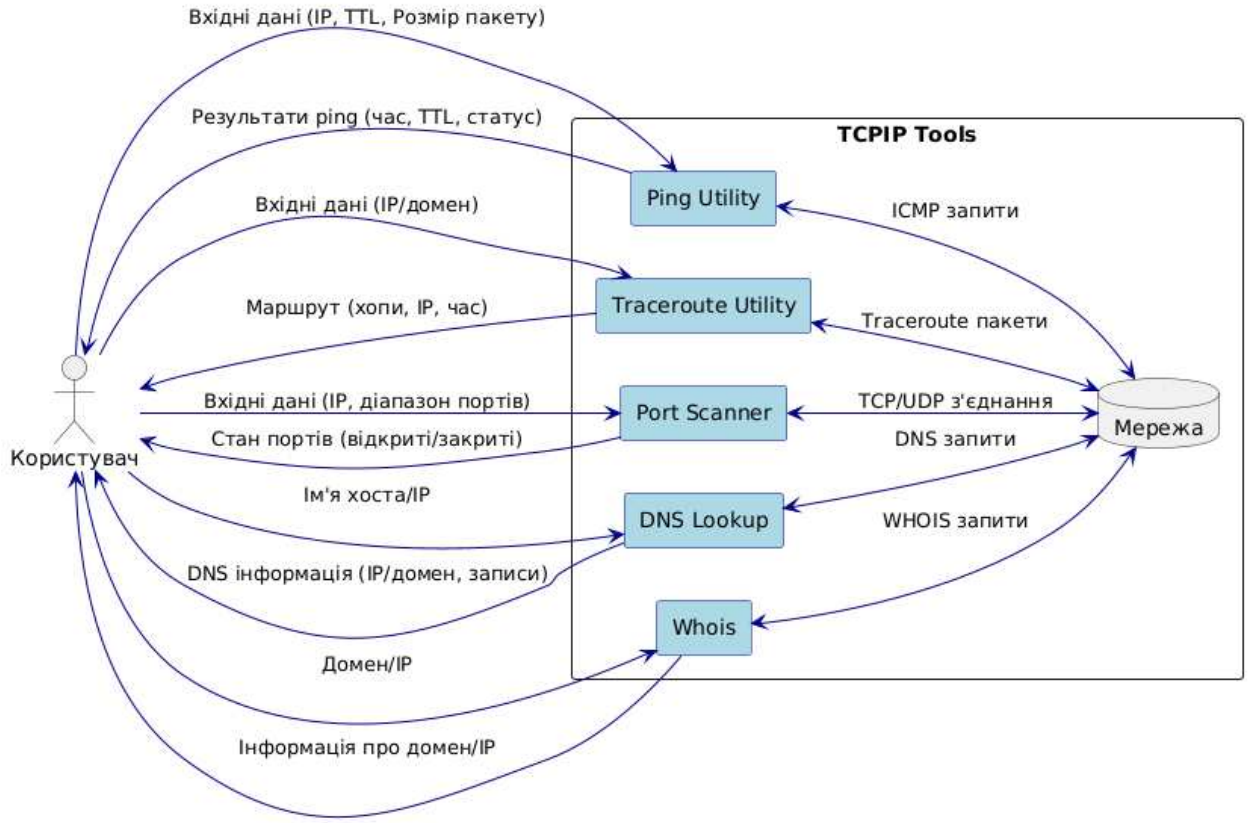

Рисунок 2.2 – Діаграма потоків даних першого рівня

DNS Lookup дозволяє користувачеві виконувати пряме або зворотне перетворення між доменним ім’ям і IP-адресою. Утиліта надсилає DNS-запити до мережі і отримує відповідні записи (A, AAAA, CNAME, тощо). Після отримання відповідей вона передає їх у зручному вигляді ініціатору запита, що дозволяє отримати інформацію про домен або IP.

Whois Utility отримує від адміністратора запит на доменне ім'я або IP-адресу. Утиліта надсилає WHOIS-запит до мережі та повертає реєстраційну інформацію про домен або IP-адресу, таку як дані про реєстратора, дати реєстрації, терміни дії та контактні дані.

Усі утиліти мають доступ до зовнішнього середовища — мережі Інтернет, що виступає джерелом та приймачем технічних запитів. Утиліти обмінюються даними через відповідні протоколи (ICMP, DNS, TCP/UDP, WHOIS), забезпечуючи ефективну діагностику стану мережі.

2.3 Алгоритм функціонування пакету TCP/IP утиліт для діагностики мережі

На рисунку 2.3 наведено блок-схему алгоритма пакету утиліт, що розробляються для діагностики мережі.

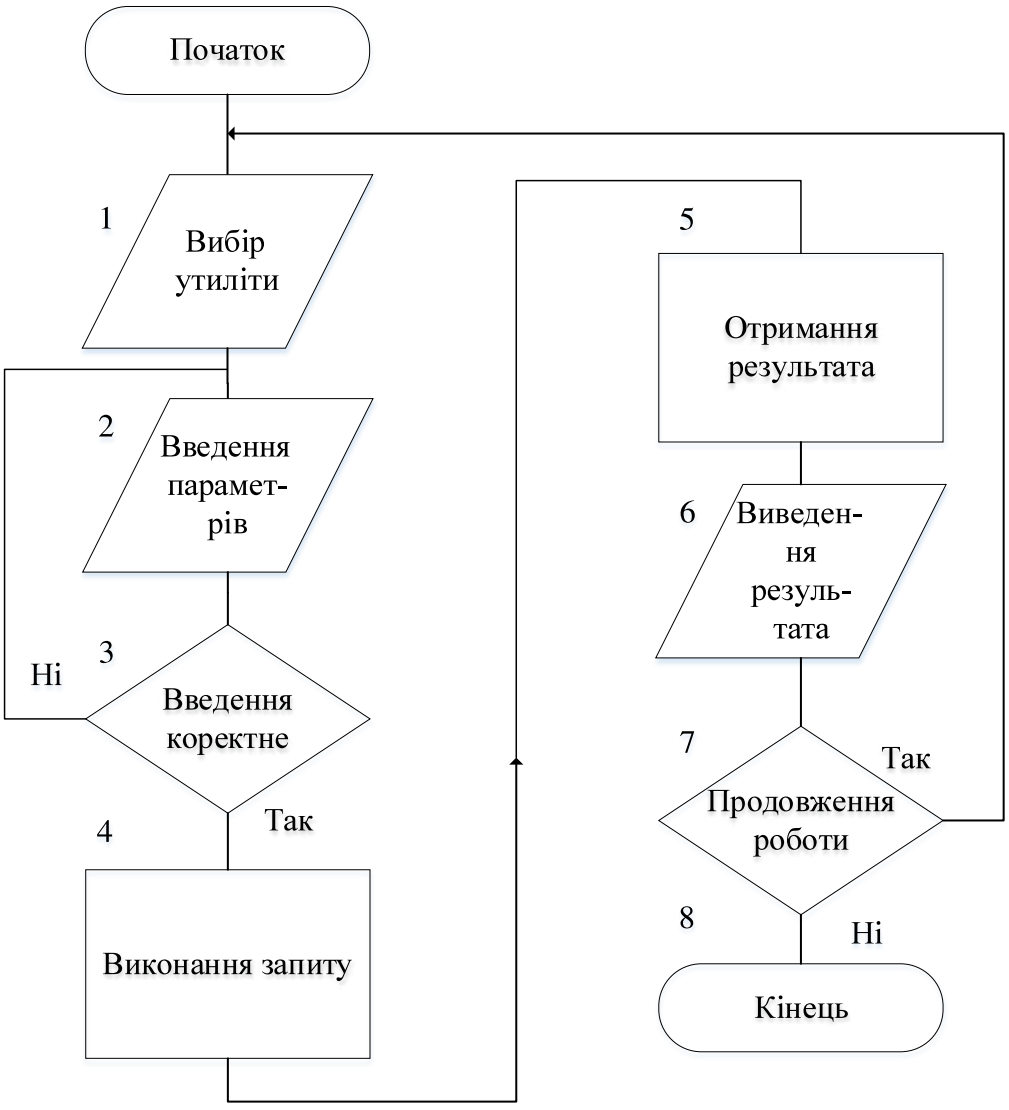

Рисунок 2.3 – Алгоритм роботи пакету TCP/IP утиліт для діагностики мережі

Програма запускається користувачем, який може вибрати одну з п’яти утиліт: Ping, Traceroute, Port Scanner, DNS Lookup, або Whois. Після вибору утиліти програма запитує у користувача необхідні дані для виконання запиту. Залежно від утиліти ці дані можуть варіюватися. Для Ping необхідно ввести IP-адресу, TTL і розмір пакета, для Traceroute — IP-адресу або доменне ім’я, для Port Scanner — IP-адресу і діапазон портів, для DNS Lookup — доменне ім’я або IP-адресу, для Whois — домен або IP-адресу.

Після того, як користувач введе дані, програма перевіряє їх на коректність. Якщо дані введені неправильно, програма інформує користувача і запитує виправлення помилки. Якщо ж перевірка проходить успішно, програма виконуватиме відповідний запит до мережі.

Для утиліти Ping програма надсилає ICMP-запит, щоб перевірити доступність хоста. Для Traceroute вона надсилає спеціальні пакети до мережі для визначення маршруту до цільового вузла. У разі Port Scanner програма намагається встановити з’єднання з хостом через зазначений діапазон портів. Утиліта DNS Lookup відправляє запит до DNS-серверів для перетворення домену в IP-адресу (або навпаки). Утиліта Whois виконує WHOIS-запит для отримання реєстраційної інформації про домен чи IP-адресу.

Після отримання відповіді від мережі програма аналізує дані і формує результат, який виводить на екран. Результати можуть включати час відгуку і статус доступності хоста для Ping, маршрут до хоста і час на кожному етапі для Traceroute, стан портів для Port Scanner, DNS записи для DNS Lookup, а також реєстраційну інформацію про домен або IP для Whois.

Якщо під час виконання запиту виникають помилки або проблеми з мережею, програма виводить повідомлення про помилку, намагаючись пояснити причину. У кінці програма завершить виконання, але користувач може повторити запит або вибрати іншу утиліту.

Алгоритм забезпечує зручний інтерфейс для користувача та дозволяє ефективно діагностувати різні аспекти мережі, включаючи доступність хостів, маршрути, стан портів, DNS-записи і реєстраційну інформацію.

3 РЕАЛІЗАЦІЯ ПАКЕТУ TCP/IP УТИЛІТ ДЛЯ ДІАГНОСТИКИ МЕРЕЖІ

3.1 Опис розробленого пакету прикладних програм

Зовнішній вигляд головного вікна створеного пакету TCP/IP утиліт для діагностики мережі наведено на рисунку 3.1. Кожна вкладка містить певний інструмент для роботи з мережею, але всі вони дотримуються єдиного стилю.

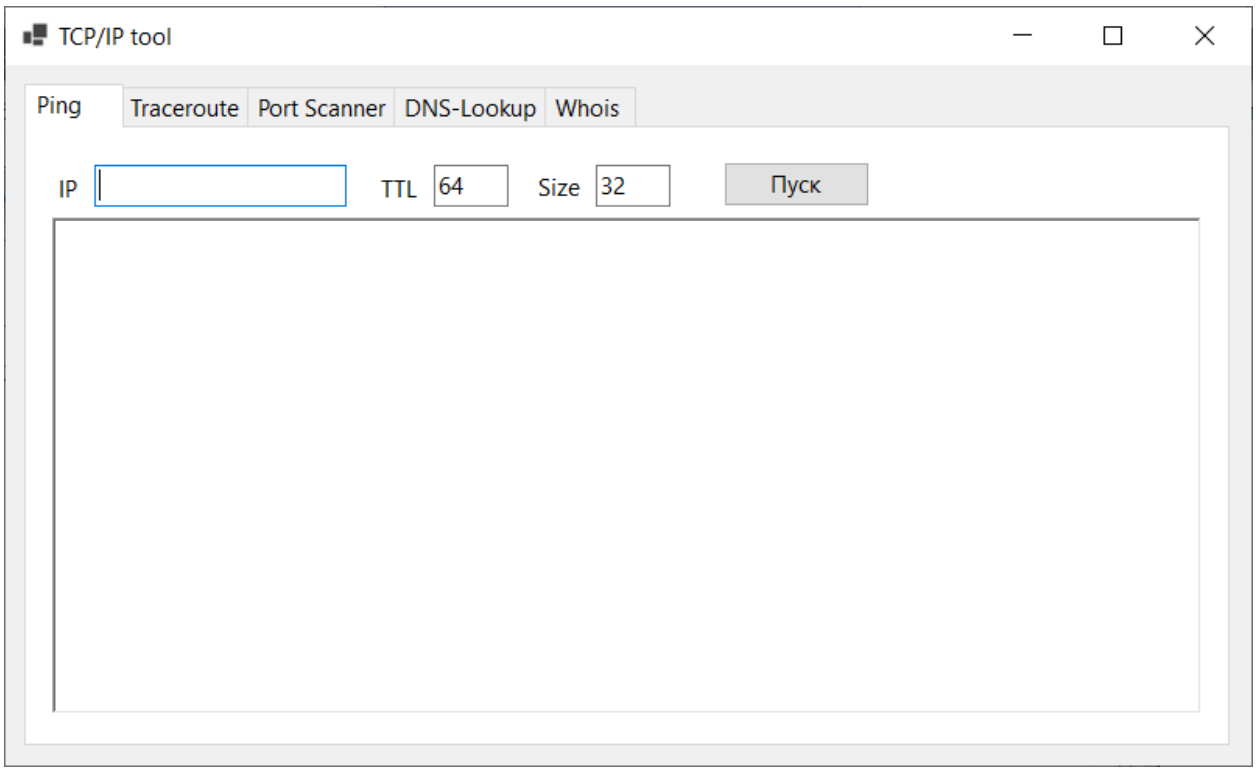

Рисунок 3.1 – Головне вікно створеного пакету TCP/IP утиліт для діагностики мережі

Верхня частина кожної вкладки містить поля для введення параметрів. Під ними розташована кнопка запуску тієї чи іншої утиліти, яка змінює свій напис під час виконання операції. Основну частину вікна займає текстове поле текстового виведення, де відображаються результати роботи обраного інструменту.

Інтерфейс використовує стандартні елементи керування Windows. Усі повідомлення про помилки або попередження з'являються у вигляді спливаючих вікон, що дозволяє користувачеві швидко зрозуміти, як виправити проблему (рисунок 3.2).

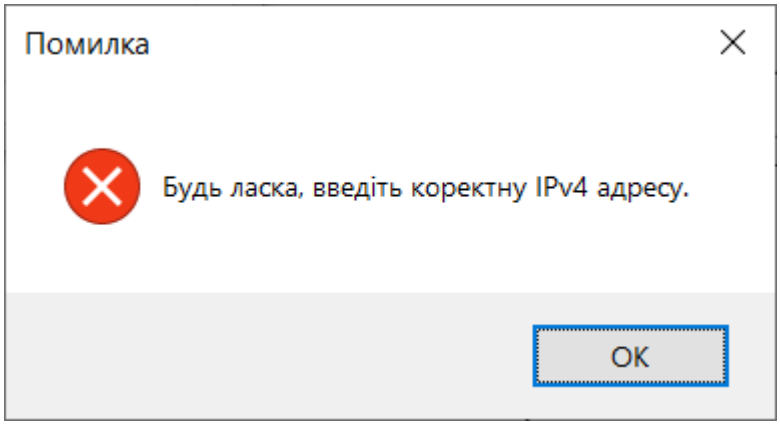

Рисунок 3.2 – Обробка помилок

Скріншот роботи програми Ping наведено на рисунку 3.3.

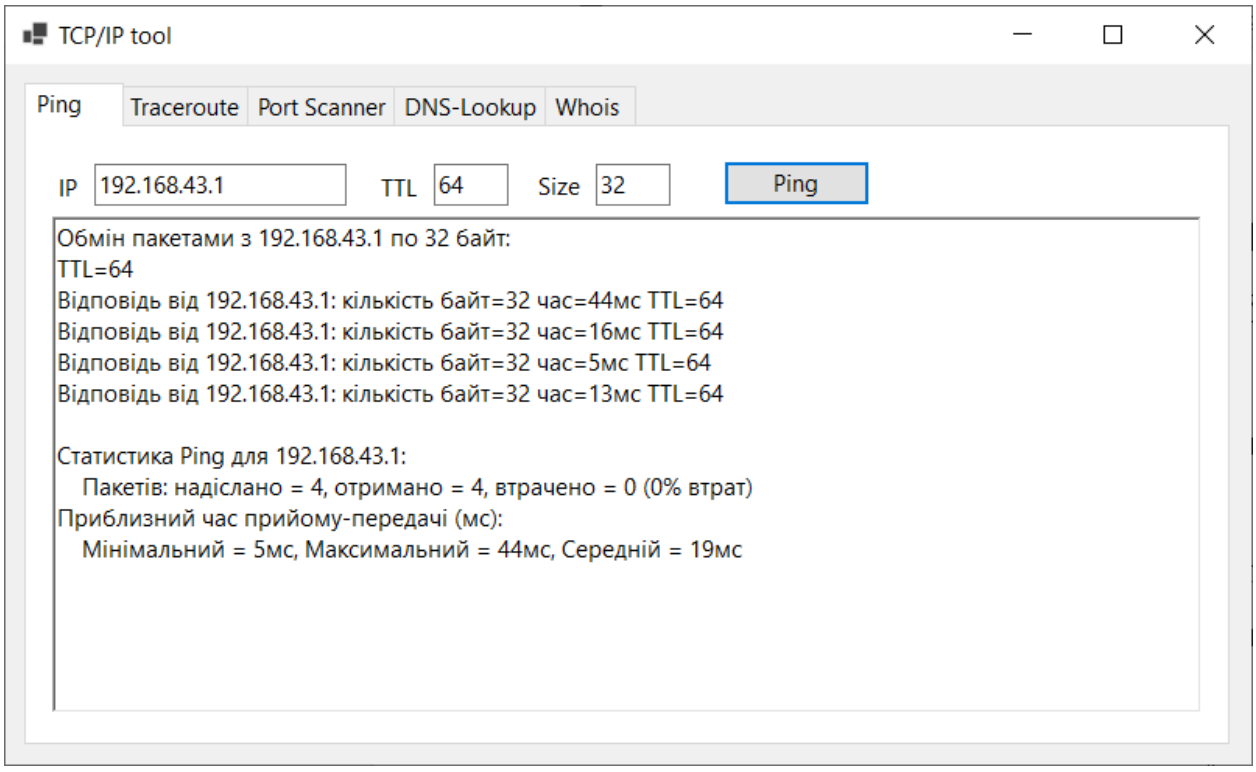

Рисунок 3.3 – Програма Ping

Утиліта реалізована у вигляді класу, який інкапсулює всю логіку роботи з ICMP-запитами. Після натискання кнопки запуску відбувається перевірка введення та створення буферу даних вказаного розміру, який заповнюється випадковими байтами та встановлюється TTL.

Запити відправляються асинхронно з інтервалом у одну секунду. Кожен запит містить ідентифікатор та послідовний номер, що дозволяє відстежувати відповіді. Під час очікування відповіді використовується Stopwatch для точної фіксації часу виконання запиту.

Отримані відповіді аналізуються на наявність помилок. У разі успіху виводиться інформація про розмір отриманого пакета, IP-адресу відправника, час відгуку та значення TTL. Якщо відповідь не надійшла протягом таймауту, це також відображається у вікні результатів.

Після відправки всіх пакетів формується зведена статистика, яка включає кількість відправлених та отриманих пакетів, відсоток втрат, а також мінімальний, максимальний та середній час відгуку.

Утиліта Traceroute використовує механізм TTL у IP-пакетах для виявлення маршруту до цільового хоста. Відправляючи пакети з послідовно зростаючим TTL, програма отримує зворотні повідомлення від кожного проміжного маршрутизатора, дозволяючи відтворити повний шлях пакетів через мережу. Процес її роботи наведено на рисунку 3.4.

Для кожного проміжного вузла відображається інформація про його місце у маршруті, IP-адресу та час відгуку. У випадку недоступності вузла або перевищення часу очікування це відзначається додатково. Також здійснюється спроба визначення імені хоста для кожної IP-адреси, що покращує розуміння шляху пакету.

Трасування завершується при досягненні кінцевого вузла або при перевищенні максимальної кількості кроків. Після завершення користувач отримує зведену інформацію про виконане трасування, що включає загальну кількість хопів та середній час відгуку.

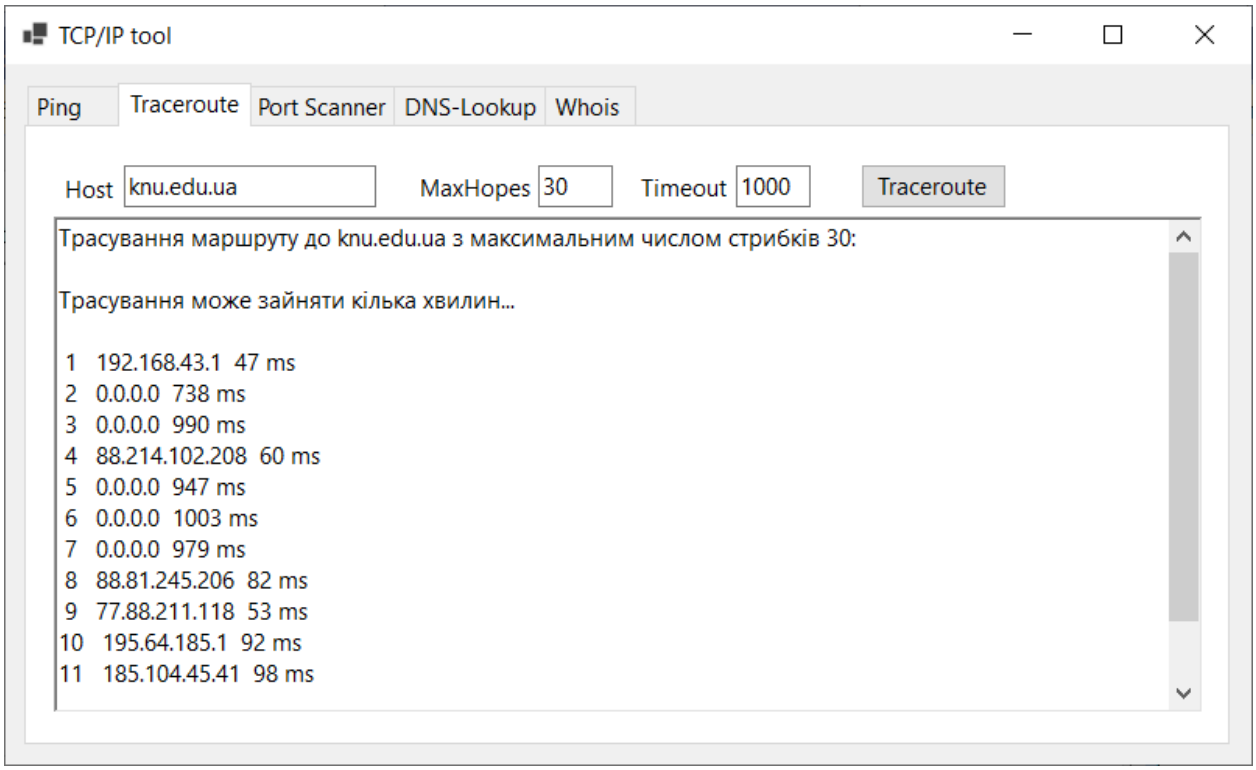

Рисунок 3.4 – Програма Traceroute

Утиліта Port Scanner призначена для виявлення відкритих портів на віддалених комп'ютерах. Вона послідовно намагається встановити TCP-з'єднання з кожним портом у вказаному діапазоні, визначаючи, які з них знаходяться у стані прослуховування. Клас PortScanner використовує асинхронні операції для одночасного сканування кількох портів, що суттєво прискорює процес роботи. Для кожного порту створюється окреме завдання, яке намагається встановити з'єднання з вказаним таймаутом. Керування кількістю одночасних з'єднань запобігає перевантаженню мережі та системних ресурсів.

Для кожного перевіреного порту відображається його номер, стан (відкритий/закритий) та назва відповідної мережевої служби. Відкриті порти логуються в звіті, а результати оновлюються в реальному часі, дозволяючи спостерігати за ходом сканування.

Програма містить вбудований словник поширених портів та відповідних їм мережевих служб. Це дозволяє надати зрозумілу інформацію про їх

призначення користувачеві. Для невідомих портів відображається лише номер відкритого порту.

Робота утиліти Port Scanner наведена на рисунку 3.5.

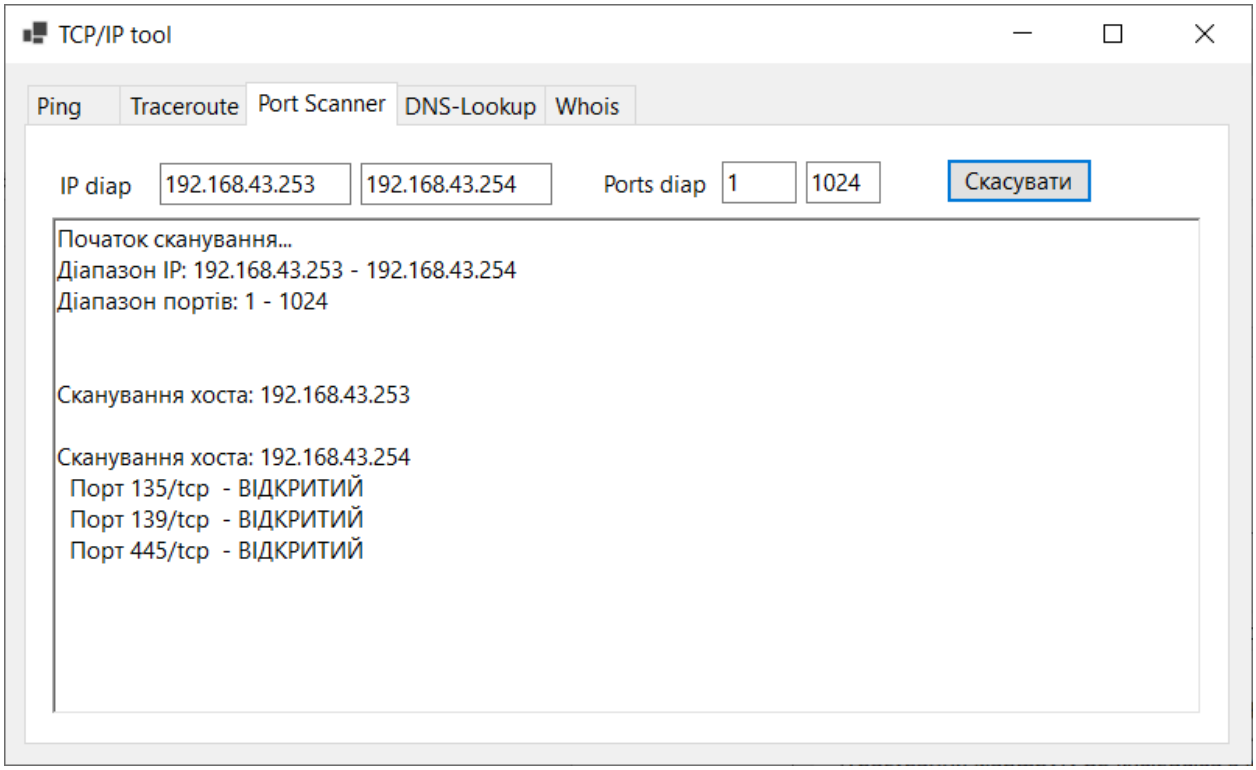

Рисунок 3.5 – Утиліта Port Scanner

Утиліта DNS Lookup призначена для отримання інформації про доменні імена та IP-адреси. Вона дозволяє досліджувати структуру DNS-записів, перетворюючи зручні для людини доменні імена у машинні IP-адреси та навпаки.

Програма аналізує вхідні дані для визначення типу запиту. В залежності від введеного значення (IP-адреса або доменне ім'я) виконується відповідний тип запиту до DNS-серверів. Для доменних імен здійснюється пошук всіх пов'язаних IP-адрес, тоді як для IP-адрес виконується зворотний пошук для отримання пов'язаних імен.

Результати запитів форматуються наступному вигляді: для кожного запиту виводиться набір доступної інформації, включаючи канонічні імена, аліаси та додаткові записи. Для доменних імен додатково здійснюється пошук

поштових серверів, що дозволяє отримати картину конфігурації електронної пошти домену (рисунок 3.6).

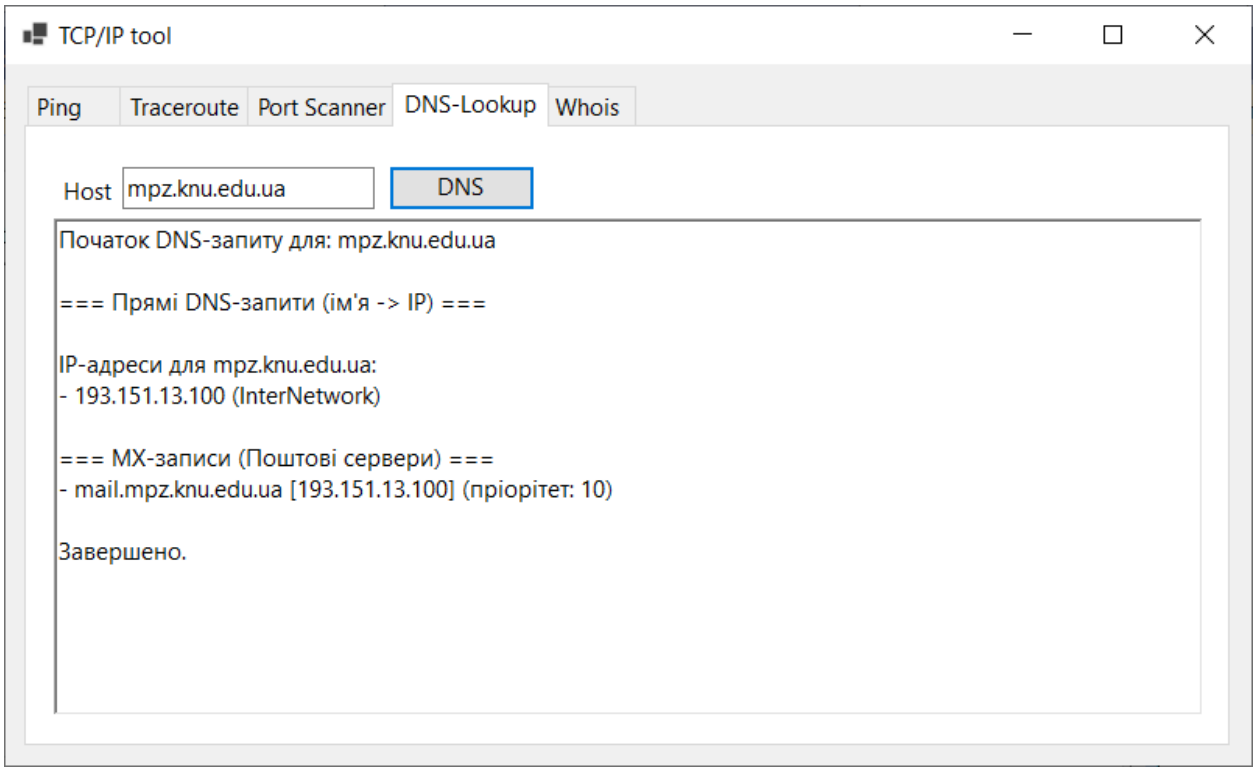

Рисунок 3.6 – Утиліта DNS Lookup

В свою чергу, утиліта Whois надає доступ до публічної інформації про доменні імена та IP-адреси, дозволяючи отримати детальні відомості про їх реєстрацію, власників та технічні параметри. Вона виконує запити до відповідних WHOIS-серверів, які зберігають дані про реєстрацію мережевих ресурсів.

Програма аналізує вхідні дані для визначення типу запиту. Для IP-адрес використовується спеціалізований сервер whois.arin.net, тоді як для доменних імен виконується пошук відповідного WHOIS-сервера на основі доменної зони.

Після встановлення з'єднання з відповідним WHOIS-сервером, утиліта надсилає запит та отримує відповідь у текстовому форматі. Відформатовані результати виводяться у текстове поле зі збереженням структури оригінальних

даних. У разі отримання посилань на додаткові джерела інформації, вони також відображаються користувачеві.

Функціонування утиліти WHOIS продемонстровано на рисунку 3.7.

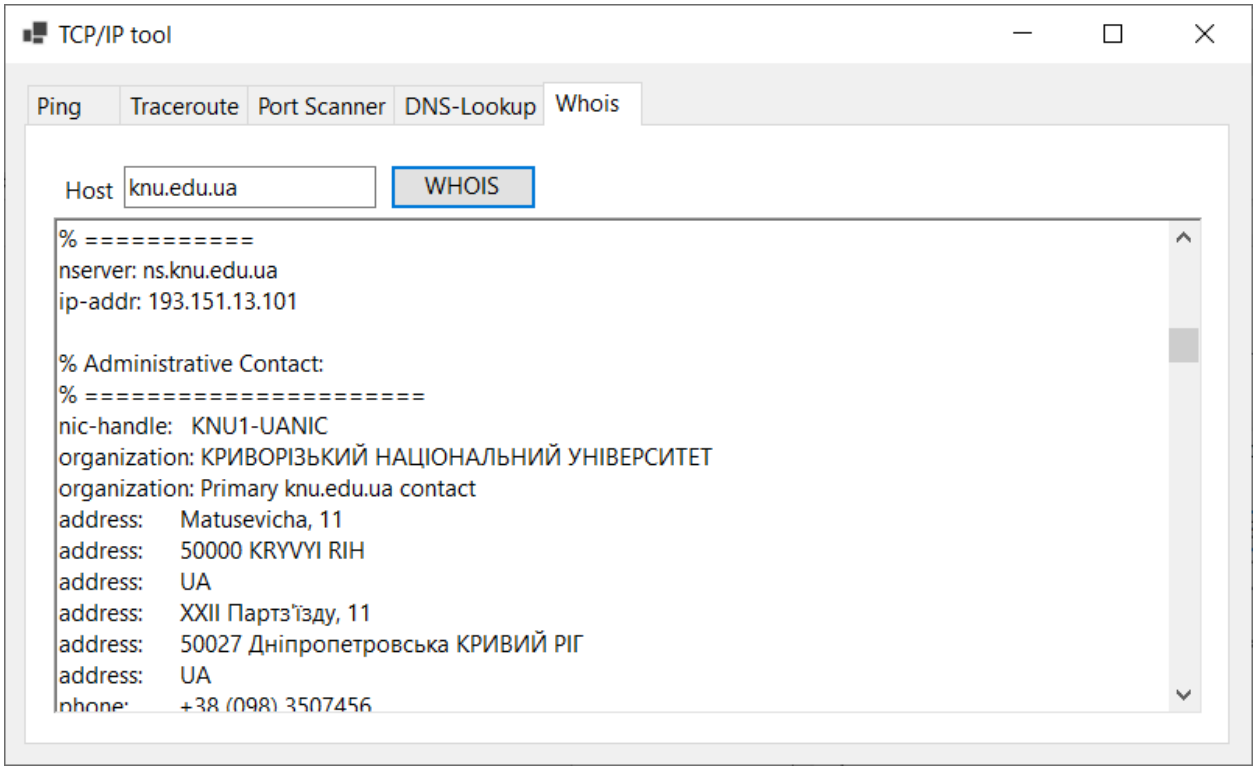

Рисунок 3.7 – Утиліта WHOIS

Таким чином, в ході роботи було створено та протестовано пакет TCP/IP утиліт для діагностики мережі, що включає в себе п’ять програм: Ping, Traceroute, Port Scanner, DNS Lookup та Whois.

ВИСНОВКИ

В ході роботи над кваліфікаційним проектом було розроблено пакет утиліт для аналізу та діагностики комп'ютерних мереж. В ньому було реалізовано п'ять основних основних програм, кожна з яких пропонує функції для роботи з мережевими протоколами.

Створений додаток демонструє ефективність при вирішенні найбільш розповсюджених завдань адміністрування мереж. Використання ефективних методологій, зокрема асинхронного програмування, забезпечило високу продуктивність та інформативність навіть при виконанні тривалих операцій. Архітектура програми, що заснована на принципах об'єктно-орієнтованого програмування, сприяє подальшому розвитку продукту. Впровадження механізмів валідації вхідних даних забезпечує стабільну роботу програми при потенційному некоректному введенні вхідних параметрів.

На основі отриманих результатів можна визначити напрямки подальшого вдосконалення програми. Перспективними напрямками розвитку можуть стати додавання підтримки IPv6, реалізація графічного відображення результатів трасування маршруту, а також розробка модулів для аналізу мережевого трафіку.

Розроблений пакет TCP/IP утиліт для діагностики мережі може використовуватися як звичайними користувачами для налаштування домашніх і робочих мереж та отримання інформації про певні домени, а також бути задіяним адміністраторами великих компаній.

ПЕРЕЛІК ПОСИЛАНЬ

1. A Comprehensive Look at the TCP/IP Model [Електронний ресурс] – Режим доступу до ресурсу: https://www.darkrelay.com/post/tcp-ip-model

2. What is: модель OSI [Електронний ресурс] – Режим доступу до ресурсу: https://rtfm.co.ua/ru/what-is-model-osi/#pll_switcher

3. Blank A. G. TCP/IP Foundations. Wiley & Sons, Incorporated, John, 2006. 304 p.

4. Задерейко О. В., Багнюк Н. В., Толокнов А. А. Комп'ютерні мережі : навчально-методичний посібник [Електронне видання]. Одеса : Фенікс, 2023. URL: https://doi.org/10.32837/11300.25951

5. Kumar A. Introduction To Computer Networks and OSI Model: Learn Yourself. Independently Published, 2020.

6. Computer Network Security / ed. by J. Rak et al. Cham : Springer International Publishing, 2017. URL: https://doi.org/10.1007/978-3-319-65127-9

7. Turay B. Analysis of Seven Layered Architecture of Osi Model. SSRN Electronic Journal. 2019. URL: https://doi.org/10.2139/ssrn.3815237

8. TCP/IP vs. OSI: What's the Difference? [Електронний ресурс] – Режим доступу до ресурсу: https://www.qsfptek.com/qt-news/tcp-ip-vs-osi-what-is-the-difference.html

9. How to use Tracert/Traceroute? [Електронний ресурс] – Режим доступу до ресурсу: https://support.n4l.co.nz/s/article/How-to-use-Tracert-Traceroute

10. What is the 'whois' command? [Електронний ресурс] – Режим доступу до ресурсу: https://www.cellstream.com/2024/03/25/what-is-the-whois-command/

11. Advanced IP Scanner [Електронний ресурс] – Режим доступу до ресурсу: https://www.advanced-ip-scanner.com

12. Essential NetTools for Windows – Режим доступу до ресурсу: https://www.tamos.com/products/legacy

13. Axence netTools [Електронний ресурс] – Режим доступу до ресурсу: https://www.majorgeeks.com/files/details/nettools.html

14. Axence netTools [Електронний ресурс] – Режим доступу до ресурсу: https://www.snapfiles.com/screenshots/axnettools.htm
15. IP-Tools [Електронний ресурс] – Режим доступу до ресурсу: https://www.ks-soft.net/ip-tools.ukr/index.htm/

Додаток А

Програмний код

PingUtility.cs

```
using System;
using System.Net.NetworkInformation;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TCPIPtool
{
    public class PingUtility
    {
        private readonly RichTextBox _outputBox;
        private readonly TextBox _ipTextBox;
        private readonly TextBox _ttlTextBox;
        private readonly TextBox _sizeTextBox;
        private readonly Button _pingButton;
        private bool _isPinging;
        private int _packetsSent;
        private int _packetsReceived;
        private long _totalTime;
        private long _minTime = long.MaxValue;
        private long _maxTime;

        public PingUtility(TextBox ipTextBox, TextBox
ttlTextBox, TextBox sizeTextBox,
                           RichTextBox outputBox, Button
pingButton)
        {
            _ipTextBox = ipTextBox;
            _ttlTextBox = ttlTextBox;
            _sizeTextBox = sizeTextBox;
            _outputBox = outputBox;
            _pingButton = pingButton;

            InitializeDefaults();
            SetupEventHandlers();
        }

        private void SetupEventHandlers()
        {
            _pingButton.Click += async (sender, e) =>
            {
                if (!_isPinging)
                    await StartPinging();
                else
                    _isPinging = false;
```

```
            };
        }


        private async Task StartPinging()
        {
            if (!ValidateInputs() || _isPinging)
                return;

            _isPinging = true;
            _pingButton.Text = "Зупинити";
            _outputBox.Clear();
            _packetsSent = 0;
            _packetsReceived = 0;
            _totalTime = 0;
            _minTime = long.MaxValue;
            _maxTime = 0;

            // Виведення заголовка
            _outputBox.AppendText($"Обмін пакетами з
{_ipTextBox.Text} по {int.Parse(_sizeTextBox.Text)} байт:" +
                                  $"\nTTL={_ttlTextBox.Text}\n");

            try
            {
                using (var ping = new Ping())
                {
                    for (int i = 0; i < 4 && _isPinging; i++)
                    {
                        await SendPing(ping, i + 1);
                        if (i < 3 && _isPinging) // Чекаємо між
пінгами, крім останнього
                            await Task.Delay(1000);
                    }
                }
            }
            catch (Exception ex)
            {
                _outputBox.AppendText($"\nПомилка:
{ex.Message}\n");
            }
            finally
            {
                if (_isPinging)
                {
                    // Виведення статистики
                    _outputBox.AppendText("\nСтатистика Ping для
" + _ipTextBox.Text + ":\n");
                    _outputBox.AppendText($"    Пакетів:
надіслано = {_packetsSent}, отримано = {_packetsReceived}, " +
                                          $"втрачено =
{_packetsSent - _packetsReceived} " +
```

```
                                            $"({(_packetsSent > 0 ?
100 * (_packetsSent - _packetsReceived) / _packetsSent : 0)}%
втрат)\n");

                    if (_packetsReceived > 0)
                    {
                        _outputBox.AppendText("Приблизний час
прийому-передачі (мс):\n");
                        _outputBox.AppendText($"    Мінімальний
= {_minTime}мс, " +
                                              $"Максимальний =
{_maxTime}мс, " +
                                              $"Середній =
{_totalTime / _packetsReceived}мс\n");
                    }
                }

                _isPinging = false;
                _pingButton.Text = "Ping";
            }
        }


        private async Task SendPing(Ping ping, int
sequenceNumber)
        {
            try
            {
                int ttl = int.Parse(_ttlTextBox.Text); //
Отримуємо TTL з текстового поля
                var options = new PingOptions
                {
                    DontFragment = true,
                    Ttl = ttl
                };

                string data = new string('a',
int.Parse(_sizeTextBox.Text));
                byte[] buffer = Encoding.ASCII.GetBytes(data);
                int timeout = 1000;

                var stopwatch =
System.Diagnostics.Stopwatch.StartNew();
                var reply = await
ping.SendPingAsync(_ipTextBox.Text, timeout, buffer, options);
                stopwatch.Stop();

                _packetsSent++;

                if (reply.Status == IPStatus.Success)
                {
                    _packetsReceived++;
```

```
                    long roundtripTime =
stopwatch.ElapsedMilliseconds;
                    _totalTime += roundtripTime;
                    _minTime = Math.Min(_minTime,
roundtripTime);
                    _maxTime = Math.Max(_maxTime,
roundtripTime);

                    _outputBox.AppendText($"Відповідь від
{reply.Address}: кількість байт={reply.Buffer?.Length} " +
                                          $"час={roundtripTime}мс
TTL={reply.Options?.Ttl ?? 0}\n");
                }
                else
                {
                    _outputBox.AppendText($"Запит перевищив
інтервал очікування.\n");
                }
            }
            catch (PingException ex)
            {
                _outputBox.AppendText($"Помилка:
{ex.InnerException?.Message ?? ex.Message}\n");
            }
            catch (Exception ex)
            {
                _outputBox.AppendText($"Помилка:
{ex.Message}\n");
            }
        }

        private bool ValidateInputs()
        {
            if (!IsValidIpAddress(_ipTextBox.Text))
            {
                MessageBox.Show("Будь ласка, введіть коректну
IPv4 адресу.", "Помилка",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
                return false;
            }

            if (!int.TryParse(_ttlTextBox.Text, out int ttl) ||
ttl < 1 || ttl > 255)
            {
                MessageBox.Show("TTL має бути числом від 1 до
255.", "Помилка",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
                return false;
            }

            if (!int.TryParse(_sizeTextBox.Text, out int size)
|| size < 1 || size > 65500)
```

```
            {
                MessageBox.Show("Розмір пакету має бути числом
від 1 до 65500.", "Помилка",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
                return false;
            }

            return true;
        }


        private bool IsValidIpAddress(string ipAddress)
        {
            string pattern = @"^((25[0-5]|2[0-4][0-9]|[01]?[0-
9][0-9]?)\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$";
            return Regex.IsMatch(ipAddress, pattern);
        }
    }
}
```

TracerouteUtility.cs

```
using System;
using System.Collections.Generic;
using System.Net;
using System.Net.NetworkInformation;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TCPIPtool
{
    public class TracerouteUtility
    {
        private readonly TextBox _ipTextBox;
        private readonly TextBox _maxHopsTextBox;
        private readonly TextBox _timeoutTextBox;
        private readonly RichTextBox _outputBox;
        private readonly Button _traceButton;
        private bool _isTracing;

        public TracerouteUtility(TextBox ipTextBox, TextBox
maxHopsTextBox, TextBox timeoutTextBox,
                                 RichTextBox outputBox, Button
traceButton)
        {
            _ipTextBox = ipTextBox;
            _maxHopsTextBox = maxHopsTextBox;
            _timeoutTextBox = timeoutTextBox;
            _outputBox = outputBox;
            _traceButton = traceButton;
```

```
            InitializeDefaults();
            SetupEventHandlers();
        }

        private void InitializeDefaults()
        {
            _maxHopsTextBox.Text = "30";    // Максимальна
кількість хопів за замовчуванням
            _timeoutTextBox.Text = "1000";   // Таймаут за
замовчуванням (мс)
        }


        private void SetupEventHandlers()
        {
            _traceButton.Click += async (sender, e) =>
            {
                if (!_isTracing)
                    await StartTraceroute();
                else
                    _isTracing = false;
            };
        }


        private async Task StartTraceroute()
        {
            if (!ValidateInputs() || _isTracing)
                return;

            _isTracing = true;
            _traceButton.Text = "Зупинити";
            _outputBox.Clear();
            _outputBox.AppendText($"Трасування маршруту до
{_ipTextBox.Text} з максимальним числом стрибків
{_maxHopsTextBox.Text}:\n");
            _outputBox.AppendText("\nТрасування може зайняти
кілька хвилин...\n\n");

            try
            {
                int maxHops = int.Parse(_maxHopsTextBox.Text);
                int timeout = int.Parse(_timeoutTextBox.Text);
                string target = _ipTextBox.Text;

                for (int ttl = 1; ttl <= maxHops && _isTracing;
ttl++)
                {
                    var result = await TraceHopAsync(target,
ttl, timeout);
```

```
                    // Форматування виводу для кожного хопу
                    string output = $"{ttl,2}   ";
                    output += result.Reply != null ?
$"{result.Reply.Address}  {result.RoundtripTime} ms" : "*    Час
очікування вийшов";

                    // Додаткові спроби, якщо перша невдала
                    for (int i = 1; i < 3 && result.Reply ==
null && _isTracing; i++)
                    {
                        result = await TraceHopAsync(target,
ttl, timeout);
                        output += result.Reply != null ? $"
{result.RoundtripTime} ms" : "    *";
                    }

                    _outputBox.AppendText(output + "\n");

                    // Перевірка, чи досягли кінцевого вузла
                    if (result.Reply != null &&
(result.Reply.Address.ToString() == target ||
                        result.Reply.Status ==
IPStatus.Success))
                    {
                        _outputBox.AppendText("\nТрасування
завершено.\n");
                        break;
                    }

                    // Додатковий відступ після кожних 5 хопів
для кращої читабельності
                    if (ttl % 5 == 0)
_outputBox.AppendText("\n");
                }

            }
            catch (Exception ex)
            {
                _outputBox.AppendText($"\nПомилка:
{ex.Message}\n");
            }
            finally
            {
                _isTracing = false;
                _traceButton.Text = "Traceroute";
            }
        }


        private async Task<(PingReply Reply, long
RoundtripTime)> TraceHopAsync(string host, int ttl, int timeout)
        {
```

```
            try
            {
                using (var ping = new Ping())
                {
                    var options = new PingOptions(ttl, true);
                    var buffer = new byte[32];
                    var stopwatch =
System.Diagnostics.Stopwatch.StartNew();
                    var reply = await ping.SendPingAsync(host,
timeout, buffer, options);
                    stopwatch.Stop();

                    return (reply,
stopwatch.ElapsedMilliseconds);
                }
            }
            catch
            {
                return (null, 0);
            }
        }


        private bool ValidateInputs()
        {
            // Перевірка IP-адреси
            if (!IsValidIpAddress(_ipTextBox.Text) &&
!IsValidHostname(_ipTextBox.Text))
            {
                MessageBox.Show("Будь ласка, введіть коректну
IPv4 адресу або ім'я хоста.",
                    "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return false;
            }

            // Перевірка максимальної кількості хопів
            if (!int.TryParse(_maxHopsTextBox.Text, out int
maxHops) || maxHops < 1 || maxHops > 64)
            {
                MessageBox.Show("Максимальна кількість хопів має
бути числом від 1 до 64.",
                    "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return false;
            }

            // Перевірка таймауту
            if (!int.TryParse(_timeoutTextBox.Text, out int
timeout) || timeout < 100 || timeout > 10000)
            {
```

```
                MessageBox.Show("Таймаут має бути числом від 100
до 10000 мілісекунд.",
                    "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return false;
            }

            return true;
        }


        private bool IsValidIpAddress(string ipAddress)
        {
            string pattern = @"^((25[0-5]|2[0-4][0-9]|[01]?[0-
9][0-9]?)\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$";
            return Regex.IsMatch(ipAddress, pattern);
        }


        private bool IsValidHostname(string hostname)
        {
            if (string.IsNullOrWhiteSpace(hostname) ||
hostname.Length > 255)
                return false;

            // Перевірка кожного сегменту хоста
            string[] segments = hostname.Split('.');
            if (segments.Length < 2)  // Мінімум 2 сегменти
(наприклад, example.com)
                return false;


            string validChars = @"^[a-zA-Z0-9][a-zA-Z0-9\-]*[a-
zA-Z0-9]|[a-zA-Z0-9]+$";
            foreach (string segment in segments)
            {
                if (string.IsNullOrEmpty(segment) ||
segment.Length > 63)
                    return false;

                if (!Regex.IsMatch(segment, validChars))
                    return false;
            }

            return true;
        }
    }
}
```

PortScanner.cs

```
using System;
```

```
using System.Collections.Concurrent;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.NetworkInformation;
using System.Net.Sockets;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TCPIPtool
{
    public class PortScanner
    {
        private readonly TextBox _startIpTextBox;
        private readonly TextBox _endIpTextBox;
        private readonly TextBox _startPortTextBox;
        private readonly TextBox _endPortTextBox;
        private readonly RichTextBox _outputBox;
        private readonly Button _scanButton;
        private bool _isScanning;
        private readonly Dictionary<int, string> _commonPorts;
        private readonly int _maxConcurrentScans = 100; //
Обмеження одночасних з'єднань
        private readonly SemaphoreSlim _throttler;

        public PortScanner(TextBox startIpTextBox, TextBox
endIpTextBox,
                           TextBox startPortTextBox, TextBox
endPortTextBox,
                           RichTextBox outputBox, Button
scanButton)
        {
            _startIpTextBox = startIpTextBox;
            _endIpTextBox = endIpTextBox;
            _startPortTextBox = startPortTextBox;
            _endPortTextBox = endPortTextBox;
            _outputBox = outputBox;
            _scanButton = scanButton;
            _throttler = new SemaphoreSlim(_maxConcurrentScans);

            // Ініціалізація словника відомих портів
            _commonPorts = new Dictionary<int, string>
            {
                { 20, "FTP-Data" },
                { 21, "FTP" },
                { 22, "SSH" },
                { 23, "Telnet" },
                { 25, "SMTP" },
                { 53, "DNS" },
                { 67, "DHCP Server" },
```

```
            { 68, "DHCP Client" },
            { 69, "TFTP" },
            { 80, "HTTP" },
            { 110, "POP3" },
            { 123, "NTP" },
            { 143, "IMAP" },
            { 161, "SNMP" },
            { 162, "SNMP Trap" },
            { 179, "BGP" },
            { 443, "HTTPS" },
            { 514, "Syslog" },
            { 520, "RIP" },
            { 587, "SMTP (SSL)" },
            { 993, "IMAP (SSL)" },
            { 995, "POP3 (SSL)" },
            { 3306, "MySQL" },
            { 3389, "RDP" },
            { 5432, "PostgreSQL" },
            { 5900, "VNC" },
            { 6379, "Redis" },
            { 8080, "HTTP-Alt" }
        };

        InitializeDefaults();
        SetupEventHandlers();
    }

    private void InitializeDefaults()
    {
        _startPortTextBox.Text = "1";
        _endPortTextBox.Text = "1024";
    }

    private void SetupEventHandlers()
    {
        _scanButton.Click += async (sender, e) =>
        {
            if (!_isScanning)
                await StartScanning();
            else
                _isScanning = false;
        };
    }

    private async Task StartScanning()
    {
        if (!ValidateInputs() || _isScanning)
            return;

        _isScanning = true;
        _scanButton.Text = "Скасувати";
        _outputBox.Clear();
```

```
            _outputBox.AppendText($"Початок сканування...\n");
            _outputBox.AppendText($"Діапазон IP:
{_startIpTextBox.Text} - {_endIpTextBox.Text}\n");
            _outputBox.AppendText($"Діапазон портів:
{_startPortTextBox.Text} - {_endPortTextBox.Text}\n\n");

            try
            {
                var startIp =
IPAddress.Parse(_startIpTextBox.Text);
                var endIp =
string.IsNullOrWhiteSpace(_endIpTextBox.Text)
                    ? startIp
                    : IPAddress.Parse(_endIpTextBox.Text);

                int startPort =
int.Parse(_startPortTextBox.Text);
                int endPort =
string.IsNullOrWhiteSpace(_endPortTextBox.Text)
                    ? startPort
                    : int.Parse(_endPortTextBox.Text);

                // Перевірка діапазону портів
                if (startPort > endPort)
                {
                    int temp = startPort;
                    startPort = endPort;
                    endPort = temp;
                }


                // Генеруємо всі IP-адреси в діапазоні
                var ipAddresses = GenerateIpAddresses(startIp,
endIp);

                // Створюємо завдання для кожного IP
                var scanTasks = new List<Task>();

                foreach (var ip in ipAddresses)
                {
                    if (!_isScanning) break;

                    var ipStr = ip.ToString();
                    _outputBox.AppendText($"\nСканування хоста:
{ipStr}\n");
                    _outputBox.ScrollToCaret();

                    // Скануємо порти для поточного IP
                    for (int port = startPort; port <= endPort
&& _isScanning; port++)
                    {
                        await _throttler.WaitAsync();
```

```
                        if (!_isScanning) break;

                        int currentPort = port; // Копіюємо
змінну для замикання
                        var scanTask = Task.Run(async () =>
                        {
                            try
                            {
                                using (var client = new
TcpClient())
                                {
                                    var connectTask =
client.ConnectAsync(ipStr, currentPort);
                                    var timeoutTask =
Task.Delay(1000);
                                    var completedTask = await
Task.WhenAny(connectTask, timeoutTask);

                                    if (completedTask ==
connectTask && connectTask.Status == TaskStatus.RanToCompletion)
                                    {
                                        // Порт відкритий
                                        string serviceName =
_commonPorts.ContainsKey(currentPort)
                                            ? $"
({_commonPorts[currentPort]})"
                                            : "";


_outputBox.Invoke((MethodInvoker)delegate {

_outputBox.AppendText($"  Порт {currentPort}/tcp {serviceName} -
ВІДКРИТИЙ\n");

_outputBox.ScrollToCaret();
                                        });
                                    }
                                }
                            }
                            catch (Exception)
                            {
                                // Порт закритий або сталася
помилка
                            }
                            finally
                            {
                                _throttler.Release();
                            }
                        });

                        scanTasks.Add(scanTask);
                    }
```

```
                }

                // Очікуємо завершення всіх завдань
                await Task.WhenAll(scanTasks);
            }
            catch (Exception ex)
            {
                _outputBox.AppendText($"\nПомилка:
{ex.Message}\n");
            }
            finally
            {
                _isScanning = false;
                _scanButton.Text = "Сканувати";
                _outputBox.AppendText("\nСканування
завершено.\n");
            }
        }

        private IEnumerable<IPAddress>
GenerateIpAddresses(IPAddress startIp, IPAddress endIp)
        {
            var startBytes = startIp.GetAddressBytes();
            var endBytes = endIp.GetAddressBytes();

            // Перевірка, чи це один і той самий IP
            if (startIp.Equals(endIp))
            {
                yield return startIp;
                yield break;
            }

            // Перевірка, чи початкова адреса менше кінцевої
            for (int i = 0; i < 4; i++)
            {
                if (startBytes[i] > endBytes[i])
                {
                    // Якщо початкова адреса більша за кінцеву,
міняємо їх місцями
                    var temp = startIp;
                    startIp = endIp;
                    endIp = temp;
                    startBytes = startIp.GetAddressBytes();
                    endBytes = endIp.GetAddressBytes();
                    break;
                }
            }

            // Генеруємо всі IP-адреси в діапазоні
            var currentIp = startIp;
            while (currentIp.Address <= endIp.Address)
            {
```

```
                yield return currentIp;

                // Збільшуємо IP-адресу на 1
                var bytes = currentIp.GetAddressBytes();
                for (int i = 3; i >= 0; i--)
                {
                    if (bytes[i] < 255)
                    {
                        bytes[i]++;
                        break;
                    }
                    else if (i > 0)
                    {
                        bytes[i] = 0;
                    }
                }
                currentIp = new IPAddress(bytes);
            }
        }

        private bool ValidateInputs()
        {
            // Перевірка початкової IP-адреси
            if (!IsValidIpAddress(_startIpTextBox.Text))
            {
                MessageBox.Show("Будь ласка, введіть коректну
початкову IPv4 адресу.",
                    "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return false;
            }

            // Перевірка кінцевої IP-адреси (якщо вказана)
            if (!string.IsNullOrWhiteSpace(_endIpTextBox.Text)
&& !IsValidIpAddress(_endIpTextBox.Text))
            {
                MessageBox.Show("Будь ласка, введіть коректну
кінцеву IPv4 адресу або залиште поле порожнім.",
                    "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return false;
            }

            // Перевірка початкового порту
            if (!int.TryParse(_startPortTextBox.Text, out int
startPort) || startPort < 1 || startPort > 65535)
            {
                MessageBox.Show("Початковий порт має бути числом
від 1 до 65535.",
                    "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return false;
```

```
            }

            // Перевірка кінцевого порту (якщо вказаний)
            if (!string.IsNullOrWhiteSpace(_endPortTextBox.Text)
&&
                (!int.TryParse(_endPortTextBox.Text, out int
endPort) || endPort < 1 || endPort > 65535))
            {
                MessageBox.Show("Кінцевий порт має бути числом
від 1 до 65535 або залиште поле порожнім.",
                    "Помилка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return false;
            }

            return true;
        }

        private bool IsValidIpAddress(string ipAddress)
        {
            if (string.IsNullOrWhiteSpace(ipAddress))
                return false;

            string pattern = @"^((25[0-5]|2[0-4][0-9]|[01]?[0-
9][0-9]?)\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$";
            return Regex.IsMatch(ipAddress, pattern);
        }
    }
}
```

DnsLookupUtility.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TCPIPtool
{
    public class DnsLookupUtility
    {
        private readonly TextBox _hostTextBox;
        private readonly RichTextBox _outputBox;
        private readonly Button _lookupButton;
        private bool _isRunning;
```

```
        public DnsLookupUtility(TextBox hostTextBox, RichTextBox
outputBox, Button lookupButton)
        {
            _hostTextBox = hostTextBox;
            _outputBox = outputBox;
            _lookupButton = lookupButton;

            SetupEventHandlers();
        }

        private void SetupEventHandlers()
        {
            _lookupButton.Click += async (sender, e) =>
            {
                if (!_isRunning)
                    await StartLookup();
                else
                    _isRunning = false;
            };
        }

        private async Task StartLookup()
        {
            if (!ValidateInputs() || _isRunning)
                return;

            _isRunning = true;
            _lookupButton.Text = "Скасувати";
            _outputBox.Clear();
            _outputBox.AppendText($"Початок DNS-запиту для:
{_hostTextBox.Text}\n\n");

            try
            {
                string host = _hostTextBox.Text.Trim();
                bool isIpAddress = IsValidIpAddress(host);

                if (isIpAddress)
                {
                    await PerformReverseDnsLookup(host);
                }
                else
                {
                    await PerformForwardDnsLookup(host);
                }
            }
            catch (Exception ex)
            {
                _outputBox.AppendText($"\nПомилка:
{ex.Message}\n");
            }
            finally
```

```
            {
                _isRunning = false;
                _lookupButton.Text = "DNS Lookup";
                _outputBox.AppendText("\nЗавершено.\n");
            }
        }

        private async Task PerformForwardDnsLookup(string
hostname)
        {
            try
            {
                _outputBox.AppendText("=== Прямі DNS-запити
(ім'я -> IP) ===\n");

                // Отримуємо всі IP-адреси для хоста
                var ipAddresses = await
Dns.GetHostAddressesAsync(hostname);
                _outputBox.AppendText($"\nIP-адреси для
{hostname}:\n");

                foreach (var ip in ipAddresses)
                {
                    _outputBox.AppendText($"- {ip}
({ip.AddressFamily})\n");
                }

                // Отримуємо канонічне ім'я хоста
                try
                {
                    var hostEntry = await
Dns.GetHostEntryAsync(hostname);
                    if
(!string.IsNullOrEmpty(hostEntry.HostName) &&
!hostEntry.HostName.Equals(hostname,
StringComparison.OrdinalIgnoreCase))
                    {
                        _outputBox.AppendText($"\nКанонічне
ім'я: {hostEntry.HostName}\n");
                    }

                    // Виводимо всі аліаси
                    if (hostEntry.Aliases != null &&
hostEntry.Aliases.Length > 0)
                    {
                        _outputBox.AppendText("\nАліаси:\n");
                        foreach (var alias in hostEntry.Aliases)
                        {
                            _outputBox.AppendText($"-
{alias}\n");
                        }
                    }
```

```
                }
                catch (Exception ex)
                {
                    _outputBox.AppendText($"\nНе вдалося
отримати додаткову інформацію: {ex.Message}\n");
                }

                // Додаткова інформація про MX-записи
                await GetMxRecords(hostname);
            }
            catch (SocketException ex)
            {
                _outputBox.AppendText($"\nПомилка DNS:
{ex.SocketErrorCode} - {ex.Message}\n");
            }
        }

        private async Task PerformReverseDnsLookup(string
ipAddress)
        {
            try
            {
                _outputBox.AppendText("=== Зворотній DNS-запит
(IP -> ім'я) ===\n");

                var hostEntry = await
Dns.GetHostEntryAsync(ipAddress);
                _outputBox.AppendText($"\nІм'я хоста:
{hostEntry.HostName}\n");

                if (hostEntry.Aliases != null &&
hostEntry.Aliases.Length > 0)
                {
                    _outputBox.AppendText("\nАліаси:\n");
                    foreach (var alias in hostEntry.Aliases)
                    {
                        _outputBox.AppendText($"- {alias}\n");
                    }
                }

                _outputBox.AppendText($"\nIP-адреси, пов'язані з
цим ім'ям:\n");
                foreach (var addr in hostEntry.AddressList)
                {
                    _outputBox.AppendText($"- {addr}
({addr.AddressFamily})\n");
                }
            }
            catch (SocketException ex)
            {
                _outputBox.AppendText($"\nПомилка зворотнього
DNS: {ex.SocketErrorCode} - {ex.Message}\n");
```

```
            }
        }

        private async Task GetMxRecords(string domain)
        {
            try
            {
                _outputBox.AppendText("\n=== MX-записи (Поштові
сервери) ===\n");

                var mxQuery = new System.Net.Mail.SmtpClient();
                var mxRecords = await Task.Run(() =>

System.Net.Dns.GetHostEntry(domain).AddressList
                        .Where(ip => ip.AddressFamily ==
AddressFamily.InterNetwork || ip.AddressFamily ==
AddressFamily.InterNetworkV6)
                        .Select(ip => $"mail.{domain} [{ip}]
(пріорітет: 10)")
                        .ToArray()
                );

                if (mxRecords.Length > 0)
                {
                    foreach (var mx in mxRecords)
                    {
                        _outputBox.AppendText($"- {mx}\n");
                    }
                }
                else
                {
                    _outputBox.AppendText("MX-записи не
знайдено.\n");
                }
            }
            catch (Exception ex)
            {
                _outputBox.AppendText($"\nНе вдалося отримати
MX-записи: {ex.Message}\n");
            }
        }

        private bool ValidateInputs()
        {
            string input = _hostTextBox.Text.Trim();

            if (string.IsNullOrWhiteSpace(input))
            {
                ShowError("Будь ласка, введіть IP-адресу або
ім'я хоста.");
                return false;
            }
```

```
            // Перевірка чи це IP-адреса
            bool isIpAddress = IsValidIpAddress(input);

            // Якщо не IP, перевіряємо чи це коректне ім'я хоста
            if (!isIpAddress && !IsValidHostname(input))
            {
                ShowError("Будь ласка, введіть коректну IPv4
адресу або ім'я хоста на латиниці.");
                return false;
            }

            return true;
        }

        private bool IsValidIpAddress(string ipAddress)
        {
            if (string.IsNullOrWhiteSpace(ipAddress))
                return false;

            string pattern = @"^((25[0-5]|2[0-4][0-9]|[01]?[0-
9][0-9]?)\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$";
            return Regex.IsMatch(ipAddress, pattern);
        }

        private bool IsValidHostname(string hostname)
        {
            if (string.IsNullOrWhiteSpace(hostname) ||
hostname.Length > 255)
                return false;

            // Дозволяємо мітки від 1 до 63 символів, що
починаються і закінчуються буквою або цифрою
            // і містять лише букви, цифри та дефіс
            string pattern = @"^([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-
Z0-9\-]{0,61}[a-zA-Z0-9])(\.([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9\-
]{0,61}[a-zA-Z0-9]))*$";

            return Regex.IsMatch(hostname, pattern);
        }

        private void ShowError(string message)
        {
            MessageBox.Show(message, "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```

WhoisUtility.cs

```
using System;
```

```
using System.Collections.Generic;
using System.IO;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TCPIPtool
{
    public class WhoisUtility
    {
        private readonly TextBox _queryTextBox;
        private readonly RichTextBox _outputBox;
        private readonly Button _lookupButton;
        private bool _isRunning;
        private const int WhoisPort = 43;
        private const int TimeoutMs = 10000;

        // Словник WHOIS серверів для різних доменних зон
        private static readonly Dictionary<string, string>
WhoisServers = new Dictionary<string,
string>(StringComparer.OrdinalIgnoreCase)
        {
            { "com", "whois.verisign-grs.com" },
            { "net", "whois.verisign-grs.com" },
            { "org", "whois.publicinterestregistry.net" },
            { "info", "whois.afilias.net" },
            { "biz", "whois.biz" },
            { "uk", "whois.nic.uk" },
            { "de", "whois.denic.de" },
            { "fr", "whois.nic.fr" },
            { "it", "whois.nic.it" },
            { "ru", "whois.tcinet.ru" },
            { "ua", "whois.ua" },
            { "eu", "whois.eu" },
            { "cn", "whois.cnnic.cn" },
            { "jp", "whois.jprs.jp" },
            { "au", "whois.auda.org.au" }
        };

        public WhoisUtility(TextBox queryTextBox, RichTextBox
outputBox, Button lookupButton)
        {
            _queryTextBox = queryTextBox;
            _outputBox = outputBox;
            _lookupButton = lookupButton;

            SetupEventHandlers();
        }
```

```
        private void SetupEventHandlers()
        {
            _lookupButton.Click += async (sender, e) =>
            {
                if (!_isRunning)
                    await StartLookup();
                else
                    _isRunning = false;
            };
        }

        private async Task StartLookup()
        {
            if (!ValidateInputs() || _isRunning)
                return;

            _isRunning = true;
            _lookupButton.Text = "Скасувати";
            _outputBox.Clear();
            _outputBox.AppendText($"Виконання WHOIS-запиту для:
{_queryTextBox.Text}\n\n");

            try
            {
                string query = _queryTextBox.Text.Trim();
                string whoisServer = GetWhoisServer(query);

                if (string.IsNullOrEmpty(whoisServer))
                {
                    _outputBox.AppendText("Не вдалося визначити
відповідний WHOIS-сервер.\n");
                    return;
                }

                _outputBox.AppendText($"Використовується WHOIS-
сервер: {whoisServer}\n\n");
                string whoisResult = await
QueryWhoisServerAsync(whoisServer, query);

                if (!string.IsNullOrEmpty(whoisResult))
                {
                    _outputBox.AppendText(whoisResult);
                }
                else
                {
                    _outputBox.AppendText("Не вдалося отримати
дані WHOIS.\n");
                }
            }
            catch (Exception ex)
            {
```

```
                _outputBox.AppendText($"\nПомилка:
{ex.Message}\n");
                if (ex.InnerException != null)
                {
                    _outputBox.AppendText($"Внутрішня помилка:
{ex.InnerException.Message}\n");
                }
            }
            finally
            {
                _isRunning = false;
                _lookupButton.Text = "WHOIS Lookup";
                _outputBox.AppendText("\nЗавершено.\n");
            }
        }

        private string GetWhoisServer(string query)
        {
            // Якщо це IP-адреса, використовуємо whois.arin.net
            if (IsValidIpAddress(query))
            {
                return "whois.arin.net";
            }

            // Якщо це доменне ім'я, визначаємо відповідний
WHOIS сервер
            string domain = query.ToLower();
            string[] parts = domain.Split('.');

            if (parts.Length >= 2)
            {
                string tld = parts[^1]; // Останній елемент -
TLD

                // Перевіряємо спочатку повний TLD (наприклад,
co.uk)
                if (parts.Length >= 3 &&
WhoisServers.TryGetValue($"{parts[^2]}.{tld}", out string
server))
                {
                    return server;
                }

                // Перевіряємо лише TLD
                if (WhoisServers.TryGetValue(tld, out server))
                {
                    return server;
                }
            }

            // За замовчуванням використовуємо whois.iana.org
            return "whois.iana.org";
```

```
        }

        private async Task<string> QueryWhoisServerAsync(string
whoisServer, string query)
        {
            using (TcpClient client = new TcpClient())
            {
                var connectTask =
client.ConnectAsync(whoisServer, WhoisPort);
                var timeoutTask = Task.Delay(TimeoutMs);

                if (await Task.WhenAny(connectTask, timeoutTask)
!= connectTask)
                {
                    throw new TimeoutException($"Час очікування
з'єднання з {whoisServer} вичерпано.");
                }

                using (NetworkStream stream =
client.GetStream())
                using (StreamWriter writer = new
StreamWriter(stream, Encoding.ASCII))
                using (StreamReader reader = new
StreamReader(stream, Encoding.UTF8))
                {
                    // Відправляємо запит
                    await writer.WriteLineAsync(query);
                    await writer.FlushAsync();

                    // Читаємо відповідь
                    StringBuilder response = new
StringBuilder();
                    char[] buffer = new char[4096];
                    int bytesRead;

                    while ((bytesRead = await
reader.ReadAsync(buffer, 0, buffer.Length)) > 0)
                    {
                        if (!_isRunning)
                        {
                            _outputBox.AppendText("\nЗапит
скасовано користувачем.\n");
                            return string.Empty;
                        }

                        response.Append(buffer, 0, bytesRead);
                        _outputBox.AppendText(new string(buffer,
0, bytesRead));
                        _outputBox.ScrollToCaret();
                        Application.DoEvents();
                    }
```

```
                    return response.ToString();
                }
            }
        }

        private bool ValidateInputs()
        {
            string input = _queryTextBox.Text.Trim();

            if (string.IsNullOrWhiteSpace(input))
            {
                ShowError("Будь ласка, введіть IP-адресу або
доменне ім'я.");
                return false;
            }

            // Перевірка чи це IP-адреса або коректне доменне
ім'я
            if (!IsValidIpAddress(input) &&
!IsValidDomainName(input))
            {
                ShowError("Будь ласка, введіть коректну IPv4
адресу або доменне ім'я на латиниці.");
                return false;
            }

            return true;
        }

        private bool IsValidIpAddress(string ipAddress)
        {
            if (string.IsNullOrWhiteSpace(ipAddress))
                return false;

            string pattern = @"^((25[0-5]|2[0-4][0-9]|[01]?[0-
9][0-9]?)\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$";
            return Regex.IsMatch(ipAddress, pattern);
        }

        private bool IsValidDomainName(string domain)
        {
            if (string.IsNullOrWhiteSpace(domain) ||
domain.Length > 255)
                return false;

            // Дозволяємо мітки від 1 до 63 символів, що
починаються і закінчуються буквою або цифрою
            // і містять лише букви, цифри та дефіс
            string pattern = @"^([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-
Z0-9\-]{0,61}[a-zA-Z0-9])(\.([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9\-
]{0,61}[a-zA-Z0-9]))*$";
```

```
            return Regex.IsMatch(domain, pattern);
        }

        private void ShowError(string message)
        {
            MessageBox.Show(message, "Помилка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```