МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Кафедра кібербезпеки та програмного забезпечення

На правах рукопису

Шевчук Ілля Іванович

Програмне забезпечення системи діагностування мережі WiFi 6

Спеціальність: 123 «Комп’ютерна інженерія»

Освітній ступінь: бакалавр

Науковий керівник:

Дрєєв Олександр Миколайович _____

(підпис) (дата)

кандидат технічних наук, доцент

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри

_____ О.А. Смірнов

(підпис) ПІБ

«____»_____2021 р.

Кропивницький – 2021

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Спеціальність *123 Комп'ютерна інженерія*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« 11 » січня 2021 року

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

Шевчуку Іллі Івановичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи діагностування мережі WiFi 6*

керівник роботи *Дрєєв Олександр Миколайович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 204-02 від 28.12.2020 року

2. Строк подання студентом роботи до захисту *22.05.2021 р.*

3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення системи діагностування мережі WiFi 6*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

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

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

6. Дата видачі завдання « 11 » січня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2021 р.	
3.	Розробка моделі компонента	20.03.2021 р.	
4.	Розробка структур даних	25.03.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2021 р.	
6.	Програмування алгоритмів	10.04.2021 р.	
7.	Оформлення ПЗ	17.04.2021 р.	
8.	Попередній захист роботи	14.05.2021 р.	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Шевчук І.І. Програмне забезпечення системи діагностування мережі WiFi 6. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній кваліфікаційній бакалаврській розроблено програмне забезпечення, яке призначено для системи діагностування мережі WiFi 6.

Метою розробки є програмне забезпечення системи діагностування мережі WiFi 6.

Результат роботи – програмна реалізація системи діагностування мережі WiFi 6.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows XP/Vista/7/8/10.

Програму розроблено в середовищі RAD Studio Delphi 10.4.

Ключові слова: комп'ютерна інженерія, мережа, WiFi 6

ABSTRACT

Shevchuk I.I. WiFi Diagnostic System Software 6. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this bachelor's qualification the software which is intended for system of diagnostics of a network WiFi 6 is developed.

The purpose of the development is the software of the WiFi 6 network diagnostic system.

The result is a software implementation of the WiFi 6 network diagnostics system.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on an IBM PC with Windows XP / Vista / 7/8/10.

The program is developed in the environment of RAD Studio Delphi 10.4.

Keywords: computer engineering, network, WiFi 6

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

1024QAM	–	Модуляція
ВОЛЗ	–	Волоконно-оптична лінія зв’язку
ПЗ	–	Програмне забезпечення
DoS	–	Атака типу «відмова в обслуговуванні»
MU-MIMO	–	Розподіл частотного ресурсу
OBSS	–	Набори базових послуг, що перекриваються
OFDMA	–	Множинний доступ з ортогональним частотним поділом
PoE	–	Power-over-Ethernet
SNR	–	Співвідношення сигнал/шум
TWT	–	Функція цільового часу пробудження
VoFI	–	Voice over Wi-Fi
Wi-Fi 6	–	Стандарт бездротових мереж 802.11ax
WLAN	–	Бездротові мережі
WTP	–	Wireless Termination Point

ВСТУП

Актуальність теми. Новий стандарт Wi-Fi 6 (802.11ax) є спадкоємцем попередніх поколінь бездротових мереж. Але тестування Wi-Fi 6 при розгортанні й обслуговуванні має особливості, які необхідно враховувати, щоб реалізувати весь потенціал нових більш дорогих пристроїв.

Мережі стандарту Wi-Fi 6 будуть впроваджуватися дуже швидко, тому що їхня поява збіглася з розгортанням мобільного зв'язку 5G і ростом попиту на високошвидкісний зв'язок для дистанційної роботи, підключених автомобілів, мереж відеоспостереження і т.д. Тому навички роботи з Wi-Fi 6 потрібні вже сьогодні.

Споконвічно стандарт Wi-Fi 6 розрахований на роботу в складному перешкодовому середовищу: у міській забудові, промислових зонах, поруч із безліччю інших бездротових мереж. Тому Wi-Fi 6 має підтримку багатоантенних приймачепередатчиків MU-MIMO.

Простіше говорячи, Wi-Fi 6 менше додав у швидкості, чому попередні зміни поколінь, але суттєво побільшав пропускну здатність мереж, де багато користувачів. Так, у мережі з 20 користувачами одна точка доступу 802.11ax забезпечує в чотири рази більшу пропускну здатність, чому 802.11ac. Тому при тестуванні найбільш актуальної буде перевірка сценаріїв, пов'язаних із пропускною здатністю й стійкістю зв'язки для безлічі пристроїв.

По суті, Wi-Fi 6 побудований на базі технологій попереднього стандарту 802.11ac. Основною відмінністю є багатостанційний доступ із частотним поділом каналів OFDMA, модуляція 1024QAM і скорочення інтервалів між частотами, що підносять.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи діагностування мережі Wi-Fi 6.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем діагностування мережі Wi-Fi 6.

– Дослідження системи діагностування мережі Wi-Fi 6.

– Програмна реалізація системи діагностування мережі Wi-Fi 6.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі діагностування мережі Wi-Fi 6.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи діагностування мережі Wi-Fi 6, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Для виконання діагностики Wi-Fi точок доступу можна застосувати традиційні іспитові стенди, які складаються із трьох секцій для устаткування.

Головне в такому стенді – програмне забезпечення, яке шукає помилки й проводить тестування на відповідність стандартам Wi-Fi. Це програмно-апаратний комплекс, який може збирати максимальну кількість даних про мережі Wi-Fi, у тому числі 802.11ax. Подібні інструменти вирішують усі необхідні завдання: створення теплової карти, перевірка зони покриття, діагностика неполадок і оптимізація мережі.

У цілому, процеси тестування залишаються колишніми, але слід урахувати деякі особливості Wi-Fi 6. У стандарту 802.11ax свої параметри тестів EVM, потужності передавачів і т.д. Важливо провести більш детальний спектральний аналіз, тому що Wi-Fi 6 має високу щільність сигналу й усе ще використовує переповнені перешкодами частоти 2,4 ГГц. Також модуляція 1024QAM вимагає передачі опорних сигналів з дуже низьким рівнем викривлень на частотах 5 ГГц із шириною каналу до 160 МГц.

Новим є більш детальна перевірка з метою переконатися, що пристрої правильно розподіляють частотні ресурси, для чого необхідно перевірити помилки синхронізації, налаштування потужності передавачів, розподіл частотного ресурсу в новій функції MU-MIMO Uplink. В Wi-Fi 6 оновлена схема роботи триггерного кадра, при якій точка доступу може одночасно ухвалювати пакети даних від кожного користувацького пристрою. У завантажених мережах Wi-Fi 6 створює підканали для великої кількості користувачів. Важливо переконатися, що середня пропускна здатність у такій системі буде достатньої для роботи. Дане завдання може зажадати багато часу, тому в мережах Wi-Fi 6 кращим є максимально автоматизоване тестування за допомогою комплексних рішень.

1.2 Область застосування

Вимоги до кабельної інфраструктури для мереж Wi-Fi 6

Важливо не випустити з уваги ріст вимоги до кабельної інфраструктури. Якщо вона не здатна забезпечити необхідну швидкість і живлення, впровадження мереж Wi-Fi 6 не принесе користі. Це особливо актуально у випадку, коли в мережі підприємства використовуються мідні проведення, а не оптоволокно.

У цей час при прокладці нових мереж або модернізації існуючих доцільно використовувати проведення з характеристиками від 10 Гбіт/сек зі смугою пропущення 500 МГц (Cat6) і краще. Також потрібна більш надійний захист від перешкод. Для організацій, яким необхідна більша пропускна здатність, кращим вибором може стати перехід до волоконно-оптичної мережі, яка зможе «переварити» великий трафік, що проходить через Wi-Fi 6.

При тестуванні існуючих мереж слід урахувати здатність кабельної лінії забезпечити працездатність мережі Wi-Fi 6. Так, кожній точці доступу Wi-Fi може знадобитися два з'єднання 10GBASE-T з кабельним з'єднанням не нижче Cat 6A. У такому випадку обов'язкова перевірка на сумарні наведення на близькому (PSANEXT) і далекому кінці (PSAACRF). Важливо не забути в першу чергу перевірити кабельні лінії на відповідність специфікаціям, перш ніж приступати безпосередньо до пошуку наведень.

Для таких перевірок призначені кабельні тестери. Тестер може працювати з мідними кабелями Cat 6A і до новітнього Cat 8, а також з ВОЛЗ, що робить прилад універсальним і придатним для перспективних мереж.

Ще один ключовий момент – перевірка достатності живлення PoE. Попереднім поколінням Wi-Fi вистачало PoE потужністю 13 Вт. Для багатоантенних точок доступу Wi-Fi 6 потрібен мінімум 30 Вт, а деяким – до 60 Вт PoE Type 3. Лінія PoE перевіряється на предмет омічної асиметрії. Потужність повинна розподілятися між провідниками PoE рівномірно, інакше сигнал на ділянці мережі Ethernet буде спотворюватися, що приведе до росту помилок і

погіршить роботу Wi-Fi. Сучасні кабельні тестери мають режим перевірки асиметрії опору, і можуть використовуватися для PoE різних типів.

Таким чином, тестування й діагностика мереж Wi-Fi 6 заснована на звичних методах і вимагає обліку лише деяких додаткових моментів. У цілому, перевірку мереж стандарту 802.11ax можна проводити помітно простіше при використанні більш складного устаткування з максимальною автоматизацією робочих процесів.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи діагностування мережі Wi-Fi 6, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми кваліфікаційної бакалаврської роботи

Ekahau Sidekick – Тестер для комплексного обстеження Wi-Fi

Два дводіапазонні Wi-Fi радіомодуля корпоративного рівня дозволяють скоротити час, затрачуваний на обстеження й усунення несправностей.

Обоє адаптера здатні виконувати зйомку всіх мереж 802.11 Wi-Fi, включаючи мережі нового покоління 802.11ax (Wi-Fi 6).

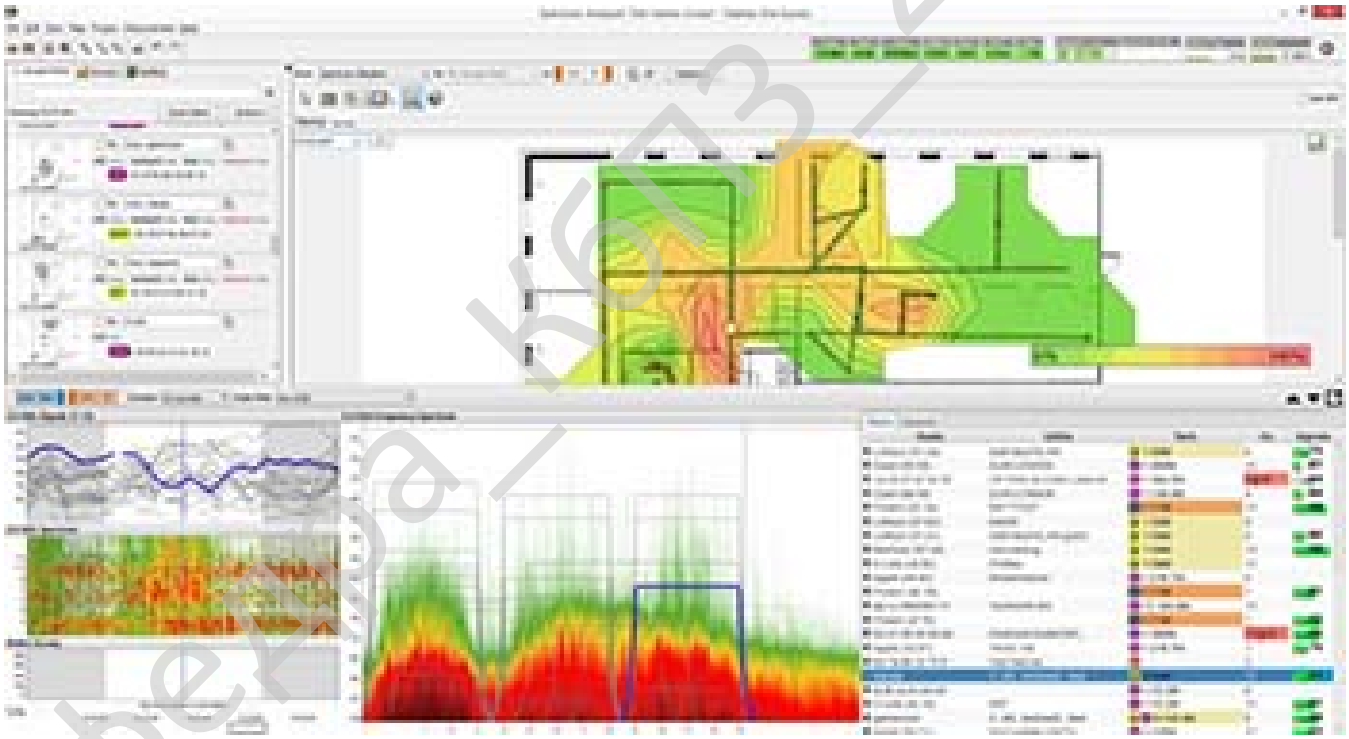

Рисунок 2.1 – Інтерфейс користувача Ekahau Sidekick

Аналізатор спектра

Аналізатор спектра надвисокої розв'язної здатності забезпечує неперевершену ясність і миттєвий аналіз перешкод Wi-Fi.

Аналізатор спектра працює в діапазонах 2,4 ГГц і 5 ГГц одночасно із частотою 20 розгорнень у секунду.

Зручність носіння

Ekahau Sidekick був розроблений, щоб носити його так, як вам подобається. Пристрій оснащений ремінцем для долоні, затискачем для ременя й має кріпильне пристосування для камери.

Ekahau Sidekick легкий і зручний у транспортуванні, що знижує стомлюваність і підвищує продуктивність.

8 годин роботи

Ekahau Sidekick має вбудований акумулятор з 8 годинами безперервної роботи, що скорочує час простою, тому проекти можна виконувати на 30% швидше.

На відміну від інших рішень, Ekahau Sidekick не буде витрачати заряд батареї вашого ноутбука або планшета.

Активні й Пасивні обстеження

У той час як Ekahau Sidekick аналізує спектр 2.4 і 5 ГГц, вимірює Wi-Fi сигнали й автоматично ідентифікує джерела перешкод, адаптер Wi-Fi на вашому ноутбуці або планшеті залишається вільним для проведення одночасних активних обстежень.

Windows, iPad або MACOS

Ekahau Sidekick – єдиний пристрій, який дозволяє проводити професійні обстеження й побудова карт покриттів Wi-Fi на Windows, iPad і MACOS.

Вибір тільки за вами.

Комплект поставки:

1. Блок Ekahau Sidekick ™.
2. Ремінець (messenger).
3. Адаптер живлення.
4. USB кабель для підключення.
5. Посібник користувача.

Ekahau Sidekick ™ – це перший у своєму роді пристрій для обстеження, перевірки, оптимізації й пошуку несправностей мереж Wi-Fi. Типовими клієнтами Ekahau Sidekick ™ є системні інтегратори й інженери корпоративних мереж у всіх галузях бізнесу, які підтримують свою бездротову мережу або проектують і розгортають мережі Wi-Fi у своїх клієнтів.

Ekahau Sidekick ™ – це пристрій з живленням від батарей, яке забезпечує більш восьми годин роботи в мережі Wi-Fi. Він поставляється із кріпильним ремінцем для перенесення, який дозволяє зручно використовувати його, не тримаючи в руках.

Оптимізоване для застосунку Ekahau Pro Site Survey

Пристрій Ekahau Sidekick ™ підключається до ноутбука або планшета, на якому встановлено програмне забезпечення Ekahau Site Survey для проектування Wi-Fi мережі. Програмне забезпечення й апаратні засоби дозволяють інженерам швидше й точніше збирати вичерпні дані про будь-яку мережу Wi-Fi, чому з будь-яким іншим розв'язком. Типові області застосування включають картографування покриття мережі Wi-Fi, аналіз, усунення неполадок і оптимізацію.

Стандартизовані виміри

До розробки Ekahau Sidekick ™, повний вимір Wi-Fi було досить складним процесом, що включають почергове використання різних інструментів, що значно збільшувало час роботи. Тепер усі обстеження можна проводити швидко й комплексно.

Два бездротові адаптери Wi-Fi 802.11ac

Вимір даних бездротової LAN містять у собі потужність сигналу Wi-Fi, шум, перешкоди, канал, інформацію про конфігурацію мережі й всебічний аналіз спектра. В Ekahau Sidekick ™ є два адаптери 802.11ac корпоративного рівня, які раніше можна було побачити тільки в корпоративних точках доступу Wi-Fi. Два дводіапазонні адаптери працюють одночасно, щоб мінімізувати час обстеження й усунення неполадок.

Зверхшвидкий високочастотний аналізатор спектра

Сьогоднішні проблеми Wi-Fi викликані не тільки Wi -Fi пристроями, але й приладами не пов'язаним з Wi-Fi, такими як мікрохвильові печі, бездротові телефони й відео передавачі. Ekahau Sidekick ™ має надвисокочастотний аналізатор спектра, який працює на частотах 2,4 і 5 ГГц одночасно. Швидкість аналізу в 4-10 раз швидше й точніше, чим в інших аналізаторів спектра Wi-Fi, що приводить до більш швидкого й точного виявлення проблем з перешкодами.

Ekahau Survey – ПЗ для iPad

Ekahau Survey for iPad – перший аналізатор Wi-Fi мережі створений спеціально для планшетів iPad. Фахівці високо цінують цей рішення, тому що планшет набагато зручніший ноутбука й дозволяє працювати від батареї значно довше.

Оптимізований для iPad

Ekahau пропонує рішення для обстеження Wi-Fi з інтуїтивно зрозумілим інтерфейсом, розроблене для надлегкого iPad™ (* iPad є зареєстрованим товарним знаком Apple Inc.)

Гарний функціонал і мала вага

Ekahau Survey для iPad робить обстеження простіше, але не заощаджує на функціях. Він автоматично знаходить усі прилеглі точки доступу й розміщає їх на карті, забезпечує миттєвий аналіз після зйомки за допомогою, що легко читаються, гарних кристально чистих теплових карт покриття.

Годинник безперервного обстеження

У комбінації з Ekahau Sidekick ™, у якому перебуває 8-година акумуляторна батарея, що перезаряджається, ви можете працювати довше без підзарядки.

Ділитеся результатами обстеження:

Ekahau Survey забезпечує безпека даних вашого обстеження Wi-Fi за допомогою функції автозберігання й дозволяє вам швидко обмінюватися

результатами опитування з досвідченими користувачами для розширеного аналізу, усунення неполадок і докладних звітів.

Системні вимоги:

– Потрібно: Apple iPad з iOS 12 або більш новий.

Рекомендовані:

– Apple iPad Pro (крім 2016 року).

– Apple iPad Air (2019)/

Ekahau Capture – захват пакетів

Легко збирайте дані, необхідні для проведення розширеного пошуку несправностей і поглибленого аналізу складних для діагностики проблем Wi-Fi. Ekahau Capture дозволяє будь-якому користувачеві швидко захоплювати пакети Wi-Fi за допомогою Ekahau Sidekick. Вам більше не потрібно вкладати гроші в дороге устаткування або використовувати складні методи для захвата пакетів.

Використовуйте два вбудовані в Ekahau Sidekick радіомодуля Wi-Fi для захвата на двох каналах одночасно, щоб не пропустити пакети, при переміщенні пристрою між двома точками доступу. Аналізуйте 2.4 і 5 ГГц одночасно.

Забезпечте безпеку мережі

Виявлення погроз безпеки, таких як неправильно настроєні параметри безпеки Wi-Fi, відкрита інформація про мережу або користувача, незашифрований трафік, атаки типу «відмова в обслуговуванні» (DoS).

Оптимізація продуктивності мереж Wi-Fi

Виявите причини високого використання каналу й розрахуйте статистику трафіку Wi-Fi, таку як швидкості повторів, середні швидкості передачі даних і розподіл типів кадрів.

Сполучимо зі сторонніми інструментами

Ekahau Capture зберігає пакети в стандартному форматі .pcap. Використовуйте будь-який аналізатор пакетів (продається окремо) для вирішення проблем з підключенням, роумінгу, перекрученого звуку, пропущених викликів і проблем із драйверами Wi-Fi.

Сполучимо з Ekahau Sidekick®

Багатофункціональний пристрій для діагностики й виміру Wi-Fi, обстеження об'єктів і аналізу спектра. Ekahau Sidekick® продається окремо.

Системні вимоги:

– Потрібно Ekahau Sidekick®.

Операційна система:

– Windows 7, 8 або 10 (64 біт).

– MACOS 10.10 або більш пізня версія.

– Процесор: 1,5 + ГГц.

– Пам'ять: 2+ ГБ ОЗП.

– Місце на жорсткому диску: потрібно не менш 1 ГБ.

Ekahau Cloud – хмарний сервіс

Працюйте в команді й аналізуйте результати разом. Виберіть метод спільної роботи, який найкраще підходить для вас і ваших клієнтів, і легко перемикайтеся між автономним і хмарним режимами. Насолоджуйтеся безперешкодним співробітництвом між центральним офісом і польовими майданчиками й робіть спільне використання проекту всією вашою командою простим і легеням. Тепер кілька людей на місцях можуть одночасно працювати над тим самим проектом, у той час як критично важливий дані можуть швидко передаватися експертам по Wi-Fi у будь-яку точку світу, щоб вони могли допомогти у вирішенні складних проблем, навіть не виходячи з офісу.

Залишайся на зв'язку

Співробітничайте й дистанційно відслідковуйте хід виконання декількох проектів практично в режимі реального часу з будь-якої точки світу.

Мінімізуй витрати часу

Запобіжите дорогим переробкам за допомогою автоматичної синхронізації проекту, яка синхронізує ваші файли між пристроями й хмарою й зберігає їх на вашому iPad, ноутбуці й Sidekick.

Організуй надійне сховище даних

Централізоване сховище проектів полегшує пошук файлів проекту й гарантує, що завжди використовується остання версія. Зберігаєте до 2500 проектів і 50 ГБ файлів проектів.

Працюй Offline при необхідності

Легко перемикайтеся між хмарним і автономним режимами, коли проекти або клієнти вимагають зберігання даних в автономному режимі.

Ekahau Connect Training – Навчання

Користуйтеся доступом до навчального контенту Ekahau Connect ™. Це навчання охоплює всі аспекти початку роботи з Ekahau Connect і його використання й служить відмінним початковим курсом для тих, хто прагне пройти більш просунуті курси Ekahau University.

Netally Airmagnet Survey – ПЗ для проектування й розгортання Wi-Fi мереж

Netally Airmagnet Survey – професійне рішення для планування й проектування локальних бездротових мереж стандартів 802.11a/b/g/n/ac. Дозволяє розрахувати ідеальну кількість, розміщення й конфігурацію AP для успішного розгортання WLAN.

Netally Airmagnet Survey доступний у двох комплектаціях «Express» і «PRO».

Airmagnet Survey Express є полегшеною версією розв'язку, що дозволяє користувачам робити основні дії по обстеженню мережі Wi-Fi з можливістю створення карти покриття, шумів і навіть продуктивності в користувацьких завданнях. Airmagnet Survey PRO розширює можливості, представлені у версії Express, і додає потужні функції: підтримку 802.11n/ac, багатоповерхове розгортання, зовнішні обстеження, перевірка проекту мережі, обстеження й підтвердження працездатності мережі, аналіз Voice over Wi-Fi (VoFI), аналіз радіочастотного спектра й багато інше.

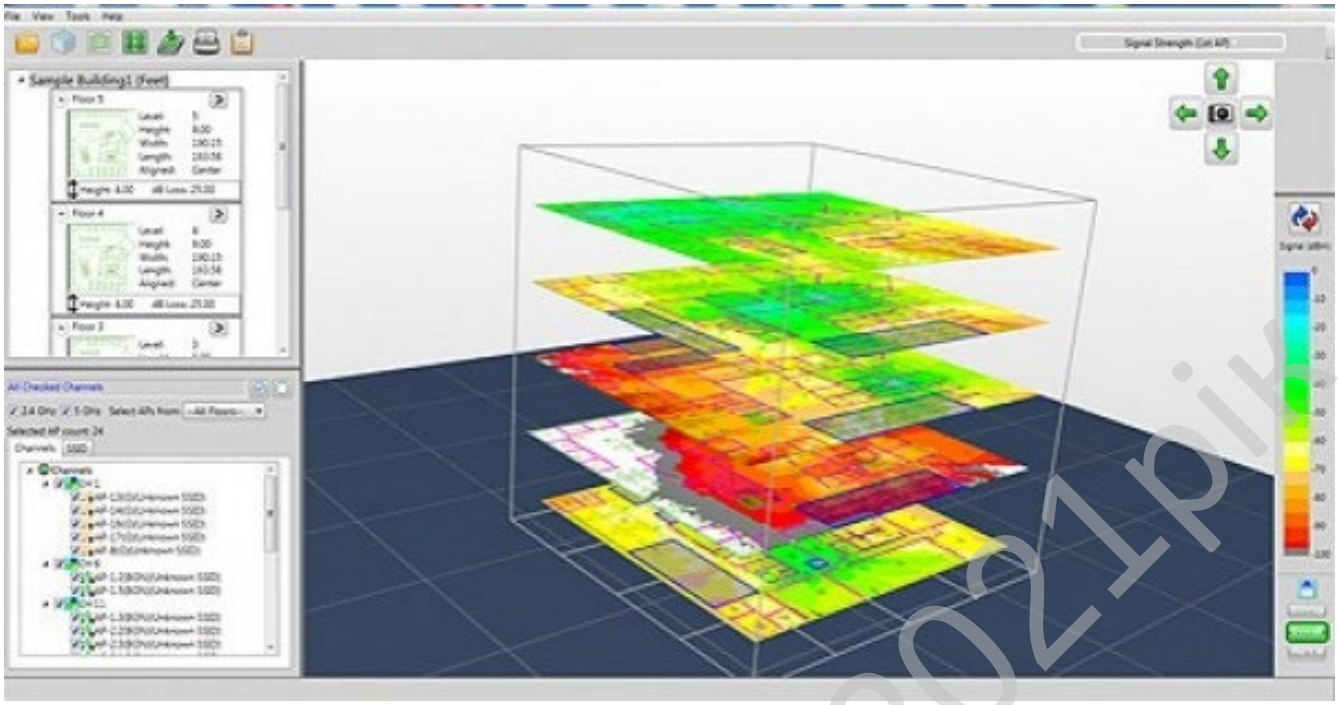

Рисунок 2.2 – Інтерфейс користувача Airmagnet Survey

Функціональні можливості Netally Airmagnet Survey:

– Обстеження ділянок внутрішніх і зовнішніх мереж стандартів 802.11a/b/g/n/ac з використанням GPS.

– Вимір реальної продуктивності у висхідному й спадному каналах.

– Аналітична функція RF Spectrum вимірює потужність і визначає місце розташування джерел перешкод.

– Карти покриття 802.11n робочого режиму, ширини каналу й продуктивності мережі.

– Вбудовані функції експертного планування ємності й перевірки проектів WLAN.

– Вбудований засіб перевірки готовності й оптимізації Wi-Fi мережі під голосовий зв'язок (VoFI).

– Що настроюються звіти про продуктивність і зоні покриття мережі.

– Підтримка імпорту планів приміщень у форматі Autocad (CAD) для автоматичного промальовування конструктивних матеріалів.

Дані можливості доступні у версії Survey PRO і частково у версії Survey Express.

Greenlee Airscout – аналізатор Wi-Fi мереж, Обстеження, Планування

Greenlee Airscout – аналізатор, що використовує інноваційний підхід до тестування, що й дозволяє провести детальну діагностику Wi-Fi мережі на етапах її інсталяції й у ході експлуатації. Він не вимагає вказівки матеріалу стін і дозволяє точно виміряти потужність сигналу у всіх точках мережі, пропускну здатність, затримку, співвідношення сигнал/шум і ін.

Tempo Airscout Gigacheck – аналізатор продуктивності Wi-Fi мережі

Артикул: GT-AGC350:

- тестування швидкості маршрутизації до 1,3 Гбіт/сек;
- визначення рівня сигналу, шуму, співвідношення сигнал/шум (SNR);
- тестування швидкості передачі й приймання Ookla®;
- визначення максимальне можливої пропускної здатності Wi-Fi мережі;
- Upload/Download Ping тестування бездротової мережі;
- тестування швидкості (Upload/Download) провідного Ethernet до 1 Гбіт/сек;
- Ethernet Ping;
- керування з користувацького пристрою (ОС Android);
- можливість зберігання результатів виміру в хмарі

Tempo Airscout 302 Residential – аналізатор Wi-Fi мережі з 2-ма віддаленими клієнтами

Артикул: GT-ASK302

- планування мережі (розміщення точки доступу, вибір каналу);
- оцінка покриття мережі, виявлення частин приміщення з недостатнім покриттям;
- оцінка пропускної здатності мережі в кожній із точок покриття;
- визначення оптимального положення точок доступу для досягнення максимально рівномірного покриття;
- рекомендації з розміщення Wi-Fi повторювачів або додаткових точок доступу;

Netally Airmagnet Spectrum XT – аналізатор спектра Wi-Fi мереж

Netally Airmagnet Spectrum XT – аналізатор спектра, який дозволяє швидко й точно робити діагностику проблем у роботі бездротових мереж (WLAN) с можливістю детального аналізу радіочастотного спектра. Збір і аналіз отриманої інформації про стан радіоефіру й вплив радіоперешкод на функціонування Wi-Fi проводиться в реальному масштабі часу й відразу виводиться на екран ПК.

Аналізатор радіочастотного спектра Airmagnet Spectrum XT дозволяє виявити й розпізнати джерела перешкод у діапазоні роботи Wi-Fi пристроїв і визначити їхнє фізичне місце розташування.

Netally Airmagnet Spectrum XT виконаний у форматі USB брелока, може бути підключений до будь-якого ноутбука/нетбуку й навіть до планшетного ПК, що має USB вхід.

Tempo Airscout LIVE PRO – аналізатор Wi-Fi мережі з аналізатором спектра

Tempo Airscout LIVE PRO – аналізатор Wi-Fi мережі. Він дозволяє швидко провести аналіз існуючої Wi-Fi мережі, визначити найбільш важливі її параметри, а також виявити й локалізувати пристрої, що впливають на пропускну здатність мережі.

Функціональні можливості аналізатора Wi-Fi Tempo Airscout LIVE PRO:

- аналіз спектра;
- аналіз використання й перевантаженості каналів;
- ідентифікація перешкод у каналах, у діапазонах 2,4 ГГц і 5 ГГц;
- пошук не Wi-Fi пристроїв, що вносять перешкоди;
- Zigbee (802.15.4) оптимізація;
- аналіз рівня сигналу користувача й точки доступу;
- визначення MAC-адреси й ідентифікація клієнтського пристрою;
- визначення впливу на канали WLAN;
- зберігання результатів виміру в хмарі (опція).

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, MAC OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для MACOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для MACOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на MACOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

– Велика кількість поліпшень STL від Dinkumware.

– Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.

– Поліпшена підтримка Cmake.

– Велика кількість виправлень для підвищення стабільності і якості.

– Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

– Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати декілька VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для MACOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для MACOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на кваліфікаційну бакалаврську роботу, реалізації підлягає програмне забезпечення, яке призначено для системи

діагностування мережі Wi-Fi 6.

В процесі розробки кваліфікаційної бакалаврської роботи необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

В останнє десятиліття усе більше сфер життя людей у переходять Інтернет, і бездротові мережі зробили такий перехід можливим. Остання версія технології бездротових мереж – технологія Wi-Fi 6, також називана 802.11ax.

Термін Wi-Fi був створений некомерційним альянсом Wi-Fi Alliance і ставиться до групи протоколів бездротових мереж, заснованих на мережному стандарті IEEE 802.11. Технологія Wi-Fi існує з кінця 90 -х, але в останнє десятиліття вона стала суттєво краще.

Таблиця 3.1 – Діаграма мережних протоколів

Покоління/стандарт IEEE	**Частота**	**Максимальна швидкість з'єднання**	**Рік**
Wi-Fi 6 (802.11ax)	2,4/5 ГГц	600–9608 Мбіт/с	2019
Wi-Fi 5 (802.11ac)	5 ГГц	433–6933 Мбіт/с	2014
Wi-Fi 4 (802.11n)	2,4/5 ГГц	72–600 Мбіт/с	2009

Щоб відмінності між поколіннями були зрозуміліше, організація Wi-Fi Alliance недавно перейшла на більш традиційний принцип присвоєння імен, замінивши формат 802.XX спрощеним числовим суфіксом. Спрощена назва (Wi-Fi 6 замість 802.11ax) дозволяє легко визначити покоління використовуваної технології й сумісність із пристроями, що підтримують цю версію.

Wi-Fi 6

Wi-Fi 6 значно перевершує технології попередніх поколінь, хоча середній користувач може не відразу помітити різницю. Ці зміни не роблять революцію в області бездротових маршрутизаторів і бездротових мереж, але містять безліч удосконалень, які в комплексі забезпечують істотне поліпшення.

Перша серйозна зміна в Wi-Fi 6 – підтримка потенційно більш високих швидкостей підключення.

Більш високі швидкості

Підвищення швидкості Wi-Fi означає підвищення швидкості вивантаження й завантаження даних (пропускної здатності) за рахунок збільшення пропускної здатності в технології Wi-Fi 6. Це дуже важливо, оскільки розміри файлів постійно ростуть, так само як і вимоги до пропускної здатності для трансляції відео у високій якості й онлайн-ігор з високим навантаженням на мережу. Щоб відіграти в багатокористувацьку гру й при цьому звістці трансляцію в Twitch*, потрібно більша пропускна здатність, а також надійне й стабільне з'єднання.

Отже, наскільки ж швидше технологія Wi-Fi 6:

– Максимальна пропускна здатність Wi-Fi 6 у багатоканальному режимі становить 9,6 Гбіт/с. Аналогічний показник Wi-Fi 5 становить не більш 3,5 Гбіт/с. Це теоретичні максимальні значення, але в реальних умовах локальні мережі можуть і не досягати такої швидкості. Однак цей максимум ділиться між усіма підключеними пристроями, так що пристрою з Wi-Fi 6 однаково стануть працювати суттєво швидше, навіть якщо й не досягнуть максимальної можливої швидкості.

– Швидкість може бути вище в порівнянні з Wi -Fi 5. Це припускає, що ви використовуєте маршрутизатор Wi-Fi з одним пристроєм. Технологія Wi-Fi 6 може забезпечити підвищення швидкості передачі даних за рахунок комбінації ряду методик, включаючи більш ефективне кодування даних і інтелектуальне використання ресурсів бездротового спектра, що стало можливим завдяки могутнішим процесорам.

– З технологією Wi-Fi 6 час затримки поменшується на величину до 75 %. Це досягається за рахунок більш ефективної обробки більших обсягів мережного трафіку. Для геймерів це означає більш швидке завантаження ігор, більш високу

швидкість вивантаження трансляцій ігрового процесу й більш надійну роботу в умовах багатозадачності.

– Технологія Wi-Fi 6 робить провідні й бездротові сигнали більш близькими по швидкості. Це потенційно може звільнити ще більше користувачів від апаратних обмежень їх модемів. Багато геймерів й автори контенту підключаються прямо до маршрутизаторів або мережних комутаторів замість того, щоб використовувати переваги гнучкості бездротових мереж. Wi-Fi 6 допомагає ще більше зменшити розрив у швидкості між провідними й бездротовими мережами.

У більшості будинків сьогодні використовується набагато більше пристроїв з підтримкою Wi-Fi, чому всього п'ять років тому. До смартфонів, планшетів і телевізорів додалися пристрої Інтернету речей, такі як термостати, дверні дзвінки й багато чого іншого. Сьогодні практично будь-який пристрій може підключатися до бездротового маршрутизатора. Технологія Wi-Fi 6 підвищує ефективність одночасного підключення декількох пристроїв і оптимізує пріоритети трафіку цих пристроїв.

Множинний доступ з ортогональним частотним поділом (OFDMA) – одна з технологій, що дозволяють добитися цього. Методика OFDMA ділить канали на декілька несучих, що дозволяє вести передачу на кілька приймачів (кінцевих пристроїв) одночасно. Маршрутизатор Wi-Fi 6 може відправляти різні сигнали в рамках одного вікна передачі. Завдяки цьому маршрутизатор може взаємодіяти з декількома пристроями в рамках одного циклу передачі, так що окремим пристроям не потрібно чекати своєї черги, поки маршрутизатор не передасть їм дані по мережі.

У традиційних мережах Wi-Fi (зверху) пристрої можуть бути змушено очікувати, поки клієнт відправить або одержить дані в мережі з більшою кількістю пристроїв. Мережі з технологією OFDMA (знизу) дозволяють обслуговувати більше пристроїв у той самий проміжок часу, що підвищує ефективність зв'язку в умовах використання великої кількості пристроїв.

Набори базових послуг, що перекриваються (OBSS) – ще одна особливість Wi-Fi 6, яка може поліпшити ситуацію в перевантажених мережах. У більш старих версіях Wi-Fi при підключенні до мережі пристрою прослуховували канал на наявність перешкод, перш ніж відправляти по ньому дані.

Якщо на каналі були присутні перешкоди, навіть від віддаленої мережі, пристроям доводилося чекати звільнення каналу від перешкод, перш ніж починати передачу. OBSS дозволяє точці доступу використовувати так звані «кольори» для унікальної ідентифікації мережі. При виявленні на каналі іншого трафіку, колір якого відрізняється від кольору локальної мережі, пристрою можуть ігнорувати його й продовжувати передачу. За рахунок цього підвищується надійність і скорочується час затримки.

Комбінація OFDMA і OBSS підвищує загальну ефективність передачі даних у завантажених мережах. Оскільки зараз усе більше пристроїв використовують Wi-Fi, це допомагає зберегти швидкість і стабільність підключень.

Формування променів – ще одна поліпшена в Wi-Fi 6 технологія, що дозволяє підвищити швидкість передачі. Цей футуристичний метод передачі даних насправді дуже простий. Замість віщання даних у всіх напрямках маршрутизатор визначає місцезнаходження запитуючого дані пристрої й направляє більш локалізований потік даних у цьому напрямку.

Звичайні маршрутизатори Wi-Fi (ліворуч) передають бездротової сигнал у всіх напрямках. Технологія формування променів (праворуч) дозволяє націлювати сигнал на кінцеві пристрої, за рахунок чого досягається більш висока швидкість підключення.

Технологія формування променів з'явилася до покоління Wi-Fi 6, однак у цім поколінні її ефективність була значно збільшена.

Переваги мережі крім швидкості

Для середнього користувача, а особливо для геймера, швидкість дуже важлива, але в бездротової мережі є й інші характеристики. Технологія Wi-Fi 6 також включає ряд поліпшень в області безпеки.

WPA3

Широко використовуваний протокол безпеки Wi-Fi Protected Access (WPA) використовує паролі для шифрування. Якщо для підключення до мережі Wi-Fi потрібно пароль, у ній використовується WPA. Стандарт WPA2 використовується вже давно, але технологія Wi-Fi 6 змінить це.

Зокрема, захист пароля підвищений за рахунок системи Dragonfly Key Exchange, що працює за принципом SAE (одночасна автентифікація рівноправних вузлів). Цей метод автентифікації ускладнює злом паролів за рахунок використання більш складного методу підключення до мережі Wi-Fi. Додатковий рівень безпеки й посилене шифрування зроблять мережі Wi-Fi надійніше, чим коли-або раніше.

Цей додатковий рівень безпеки – відмінний приклад позитивних змін в Wi-Fi 6 без негативного впливу на зручність роботи користувачів.

Час автономної роботи й TWT

Також в Wi-Fi 6 реалізована функція цільового часу пробудження (TWT) можливість, що дає, збільшення часу автономної роботи деяких пристроїв.

Дана технологія підвищує ефективність взаємодії маршрутизатора й пристрою відносно переходу в режим сну або пробудження. З'єднання з радіопередавачем Wi-Fi на пристрої активується тільки тоді, коли пристрій повинний бути активно, і тому пристрій витрачає менше часу й енергії на пошук бездротового сигналу.

Це дає можливість збільшення часу автономної роботи.

Wi-Fi 6E

На додаток до Wi-Fi 6 недавно з'явилася нова технологія Wi-Fi: Wi-Fi 6E.

Раніше пристрої Wi-Fi могли використовувати тільки частотні діапазони 2,4 ГГц і 5 ГГц, але недавно це змінилося. Пристрою з підтримкою Wi-Fi 6E можуть використовувати частотний діапазон 6 ГГц із шириною смуги пропущення 1200 МГц, що робить його ідеальним для передачі більших обсягів даних на невеликі відстані. Це може допомогти знизити перевантаження трафіку й взаємні перешкоди підтримуваних пристроїв. Wi-Fi 6E можна представити як додавання нової широкої смуги на двополосну магістраль Wi-Fi з усіма перевагами Wi-Fi 6.

Не кожний пристрій з підтримкою Wi-Fi 6 буде підтримувати Wi-Fi 6E, так що обов'язково перевірте, чи підтримує обране вами апаратне забезпечення технологію Wi-Fi 6E, перш ніж здобувати його.

От які фактори потрібно враховувати при переході на Wi-Fi 6:

– Чи підтримує ваш маршрутизатор Wi-Fi 6 Маршрутизатор з підтримкою Wi-Fi 6 – це найважливіше відновлення, необхідне для повного використання можливостей нового протоколу. Більшість виробників уже пропонують досить широкий вибір маршрутизаторів з Wi-Fi 6.

– Чи підтримують ваші пристрої Wi-Fi 6 Також вам будуть потрібні пристрої, здатні використовувати Wi-Fi 6. Хоча стандарт Wi-Fi 6 має зворотну сумісність зі старим стандартом 802.11ac (Wi-Fi 5), для одержання всіх вищеописаних переваг вам буде потрібно пристрій з підтримкою Wi-Fi 6. У найближчі кілька років Wi-Fi 6 стане стандартним розв'язком бездротового зв'язку й буде входити в базову комплектацію більшості пристроїв.

– Чи підтримує ваш ПК Wi-Fi 6 Якщо ви використовуєте процесор Intel останнього покоління й сумісну системну плату, у вас уже є багато необхідні для роботи Wi-Fi 6 компоненти, і вам буде дуже легко використовувати нову технологію. Хоча зовнішня антена M.2 усе ще необхідна, якщо її немає у вашій системній платі, реалізація технології Intel® Integrated Connectivity (Cnvi) означає, що ваша система вже містить основний технологічний фундамент Wi-Fi 6.

Якщо вас цікавить рішення Wi-Fi 6 для ігор з гігабітними швидкостями, гляньте на цю карту Killer® Wi-Fi 6 AX1650 Wi-Fi 6, розроблену разом з Intel. Ці пристрої використовують канали 160 МГц, що підтримують бездротовий зв'язок на швидкості до 1700 Мбіт/с, що в три рази перевершує стандартну швидкість Wi-Fi 5 в ідеальних умовах.

Пам’ятайте, що всі пристрої й компонента, сумісні зі стандартом Wi-Fi 6 або Wi-Fi 6E, будуть мати відповідне маркування, так що звертайте увагу на це позначення при виборі.

Wi-Fi 6 і Wi-Fi 6E суттєво поліпшать нашу взаємодію з бездротовими пристроями. Висока швидкість, поліпшена установка пріоритетів трафіку й додаткова безпека роблять Wi-Fi 6 значним кроком уперед по шляху розвитку технологій бездротових мереж.

Перехід на Wi-Fi 6 варто розглянути для ігор, роботи й навіть просто для трансляції відео.

3.2 Розробка структурної схеми

Бездротова мережа Wi-Fi забезпечує зручність і мобільність у переміщенні, при цьому швидкість передачі даних майже не уступає провідним мережам. Працюючи на ноутбуці за бездротовою технологією, можна вільно переміщатися по всій території й при цьому залишатися на зв'язку з усіма Інтернет сервісами, а також підтримувати бездротовий зв'язок з іншими робочими станціями.

Основною перевагою бездротових мереж Wi-Fi є простота їх розгортання. Також вони використовується при неможливості прокладки кабелю. Монтаж Wi-Fi можна робити в будинках будь-яких розмірів, з будь-якою кількістю кімнат. При цьому вартість устаткування й самого монтажу буде невисокою, у відмінності від аналогічного проведення кабелів в усі приміщення будинку.

Бездротовий зв'язок забезпечується в радіусі 80 – 200 метрів від точки доступу на відкритій місцевості.

Кожна мережа проектує під ті завдання, які коштують перед співробітниками й компанією в цілому. У кожному конкретному проекті підбирається оптимальний рішення з урахуванням вимог клієнта й можливостей уже існуючого устаткування.

Етапи проектування й побудови Wi-Fi мережі:

1. Створення плану об'єкта, що визначає параметри, що впливають на зону покриття.

2. Визначення числа майбутніх користувачів мережі, зони покриття, продуктивності мережі, вимог безпеки, устаткування.

3. Вибір оптимального місця для установки точок доступу.

4. Розробка умов для підключення устаткування до мережі.

5. Складання остаточної специфікації необхідного устаткування.

6. Складання кабельних схем для підключення точок доступу.

7. Складання робочої документації.

8. Установка й налаштування мережного устаткування.

9. Тестування Wi-Fi мережі, пошук факторів, що знижують якість роботи мережі.

Рівень підключення організується на базі недорогих точок доступу WTP (Wireless Termination Point), завдання яких полягає в шифруванні даних у радіоканалі й взаємодії з контролером доступу. Для підключення «тонких» точок доступу часто використовуються провідні лінії, у тому числі мережі Ethernet з підтримкою технології живлення PoE (Power-over-Ethernet).

Централізація керування всією мережею дозволяє знизити експлуатаційні витрати, автоматизувати рутинні процеси по відновленню програмного забезпечення й налаштувань точок доступу. Крім того, забезпечується високий рівень безпеки мережі, оскільки на точках доступу не зберігається яка-небудь важлива конфіденційна інформація. Ще одна істотна перевага мережі із

централізованою архітектурою полягає в тому, що при переході від однієї точки доступу до інший користувач не втрачає з'єднання з мережею, і йому не доводиться проходити автентифікацію заново.

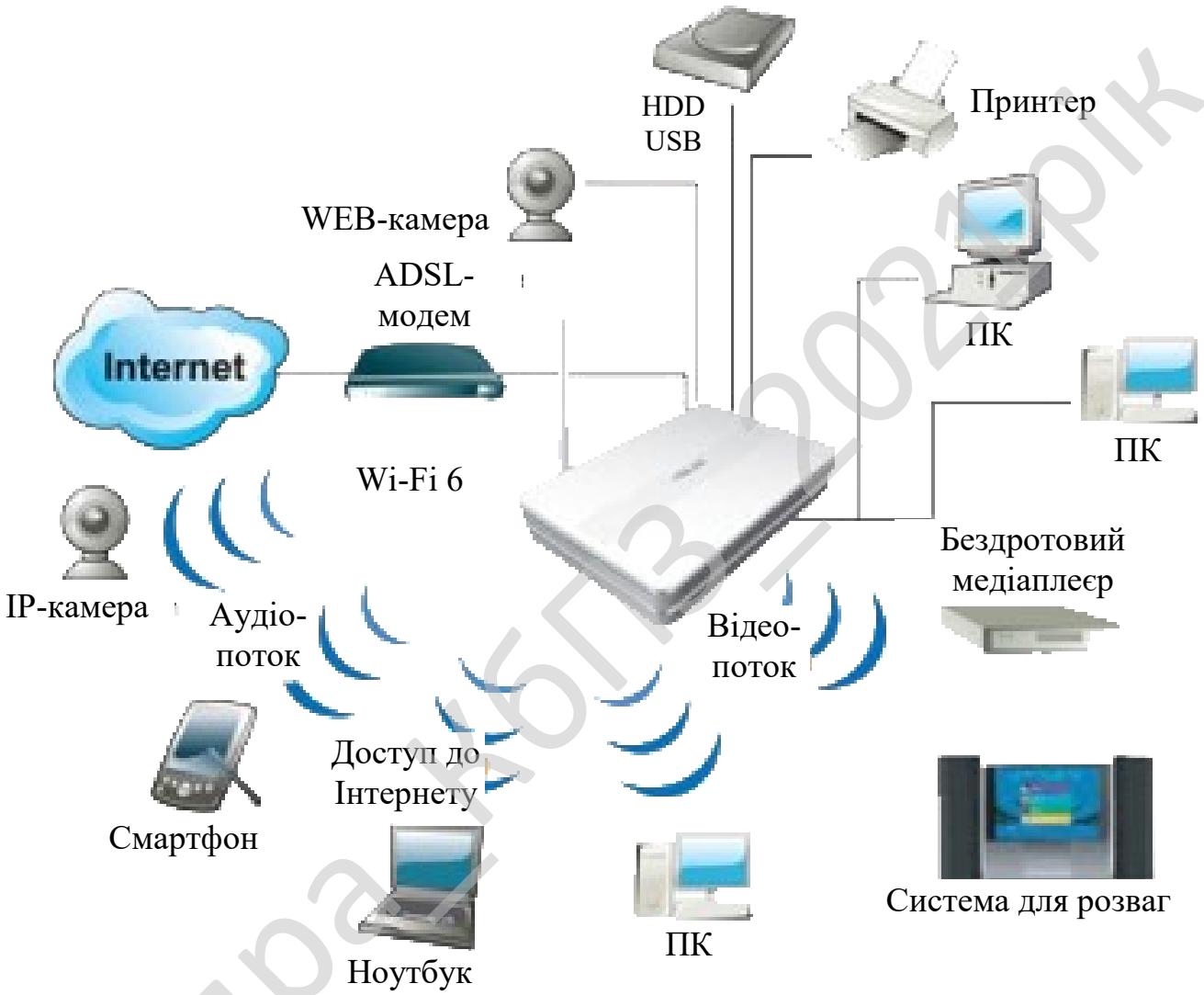

Рисунок 3.1 – Структурна схема системи

Оскільки багато точок доступу підтримують режим живлення PoE, центральний комутатор здатний не тільки забезпечити для них живлення, але й виявляти збійні ділянки мережі. Більше того, центральний комутатор може ефективно розподіляти завантаження каналів, виділяючи більш високу пропускну

здатність сегментам мережі, що нараховують у цей момент більша кількість користувачів.

Рішення проблеми формування променя

Стандартна точка доступу передає сигнал в усі сторони з рівною силою, промені розходяться по приміщенню рівномірно. Антени спрямованого дії фокусують сигнал, у результаті він при рівній потужності здатний подолати суттєво більші відстані, чому при використанні ненаправлених антен. Однак спрямовані антени мають сенс тільки в тому випадку, якщо клієнтський ПК перебуває в одному місці. Адже якщо ціль вийде за межі зони приймання, то відразу втратить сигнал

Щоб розв'язати цю проблему, використовуються системи формування променя, іменовані масивами Wi-Fi. Вони поєднують в одному корпусі безліч (від 6 до 24) різнонаправлених невеликих антен. Далі за допомогою програмного забезпечення в реальному часі визначається та комбінація антен, при якому прийнятий сигнал досягає найвищої якості. При переміщенні клієнтського ПК або іншій зміні ситуації проводиться динамічна перебудова. Така технологія Beam Forming надає відразу дві переваги. По-перше, фокусування сигналу суттєво збільшує дальність дії в порівнянні зі звичайною круговою антеною. Крім того, спрямована передача сигналів дозволяє усунути інтерференції між гніздами бездротової мережі, що позитивно позначається на пропускній здатності.

Застосування масивів Wi-Fi доцільно практично в будь-якому середовищі, але особливо його плюси проявляються там, де клієнти часто переміщаються. Як приклад можна привести фірмову технологію від компанії Ruckus Wireless, яка оптимізована для застосування в аеропортах. Через різноманітність використовуваних електронних пристроїв у цих середовищах виникає особливо багато інтерференцій, у той час як за допомогою Beamflex можна створювати дуже надійні й продуктивні мережі

3.3 Розробка функціональної схеми

Розроблене в даній роботі програмне забезпечення Diagnosing Network Wi-Fi 6 допоможе Вам спланувати, побудувати й модернізувати Wi-Fi інфраструктуру. Легкі для розуміння карти покриттів і швидкі звіти, зроблять оптимізацію бездротових мереж Wi-Fi 6 набагато простіше.

Diagnosing Network Wi-Fi 6 дозволяє планувати Wi-Fi 6 мережі, відповідно до Ваших вимог продуктивності й масштабованості. Кількість бездротових клієнтів стрімко росте. Їм потрібна більша швидкість і стабільність зв'язку для роботи з новими застосунками, VoIP, потоковим HD відео й просто для перегляду Web сторінок.

Функціональні можливості Diagnosing Network Wi-Fi 6:

– Планування покриття Wi-Fi 6.

– Типи обстеження Wi-Fi 6: Пасивні, Активні, Пропускна здатність, Спектр.

– Теплові карти покриття Wi-Fi 6.

– Пошук і вирішення проблем Wi-Fi 6.

– Звіти про тестування.

– Покриття й завантаженість Wi-Fi 6.

– Підтримка 802.11ax (Wi-Fi 6).

– Боротьба з перешкодами.

– Сумісність із операційними системами.

Планування покриття Wi-Fi 6:

Diagnosing Network Wi-Fi 6 допоможе: правильно розмістити й настроїти точки доступу Wi-Fi, спланувати покриття Wi-Fi мережі, попередньо оцінити продуктивність і пропускну здатність. В 3D форматі.

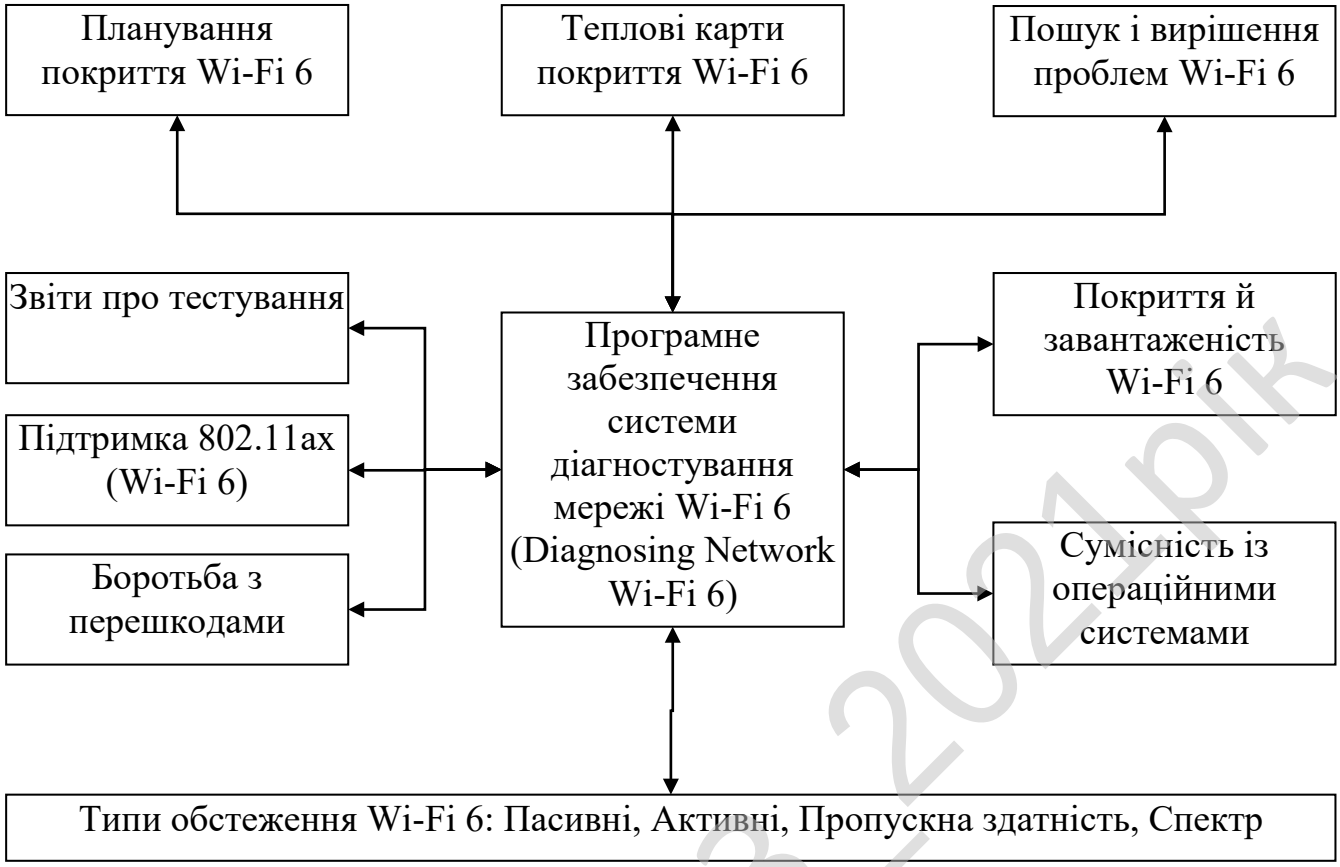

Рисунок 3.2 – Функціональна схема системи

3D-моделювання Wi-Fi 6 мережі за хвилини:

– Потужний движок для 3D моделювання.
– Багатоповерхове планування Wi-Fi 6.
– Планування покриття декількох будинків.
– Планування міських районів.
– Вибір матеріалів будівлі й висоти стін.
– Перетворення креслень із CAD у стіни.
– Облік спрямованості антен.

Автоматична установка точок доступу (ТД):

– Розміщення точки доступу одним натисканням.
– Наочні схеми покриття Wi-Fi 6 мережі.
– Більш 2500+ точок доступу Wi-Fi 6 і антен.
– Розширене планування.

– Моделювання й симуляція.

– Автоматичне розміщення й призначення каналу для точки доступу на основі вимог до покриття й пропускної здатності.

Типи обстеження Wi-Fi 6: Пасивні, Активні, Пропускна здатність, Спектр

Комбінуйте кілька варіантів тестування: пасивні й активні обстеження, оцінка пропускної здатності й аналіз спектра:

– Підтримка стандарту 802.11 a/b/g/n/ac/ax.

– Одночасні обстеження 2.4 і 5 ГГц.

– Пропускна здатність (Iperf).

– Підтримка декількох адаптерів.

– Вуличні обстеження з підтримкою GPS.

– Оптимізація роботи на сенсорному екрані.

– Знаходить усі точки доступу.

– Заповнює пробіли в місцях, які ви не відвідали.

– Автоматична оптимізація каналу.

– Одночасні пасивні й активні обстеження й аналіз спектра.

Теплові карти покриття Wi-Fi 6

Аналізуйте зібрані дані, використовуючи одну з 15 видів теплових карт покриття, щоб швидко визначити працездатність і продуктивність Wi-Fi 6 мережі:

– Ступінь покриття сигналу.

– Швидкість передачі даних.

– Втрата пакетів.

– Відношення сигнал-шум.

– Час проходження сигналу.

– Кількість точок доступу (перекриття).

– Спектральна потужність каналу.

– Ширина Wi-Fi 6 каналу.

– Джиттер.

– Пропускна здатність.

– Максимальна пропускна здатність каналу.

– Проблеми із пропускною здатністю мережі.

– Використання спектра.

– Перешкоди в сполученому / сусідньому каналі.

– Проблеми із пропускною здатністю (надмірні виклики VoIP на радіо, перевантаження точки доступу).

Пошук і вирішення проблем Wi-Fi 6

Візуальний аналіз на основі теплових карт покриття й ефективні методи усунення неполадок. Моніторинг частоти в режимі реального часу виводить діагностику на новий рівень, поєднуючи дані спектра й Wi-Fi 6 у єдиний зручний звіт. Diagnosing Network Wi-Fi 6 визначить чому ваша мережа виходить із ладу й де саме.

Пробіли в покритті Wi-Fi 6:

– Проблеми з VoIP (тремтіння, втрата пакетів).

– Надмірний трафік VoIP або перевантаження ТД.

– Несанкціоновані точки доступу.

– Відсутні налаштування безпеки.

– Надмірні перешкоди (Wi-Fi 6 і не-Wi-Fi 6).

– Проблеми із пропускною здатністю.

– Недостатня ємність Wi-Fi 6.

– Субоптимальні призначення каналів.

– Невірно настроєні точки доступу.

– Надмірна кількість клієнтів на ДТ.

– Надмірне навантаження на даний канал.

Звіти про тестування Wi-Fi 6

Аналізатор Wi-Fi 6 мережі Diagnosing Network Wi-Fi 6 створює зведений звіт з комплексом вимірів, картами покриття й продуктивності, інформацією про пропускну здатність і моделлю всієї мережі.

Одним натисканням кнопки мережа Wi-Fi 6 на всіх поверхах будинку й навіть у декількох будинках відразу буде повністю описана в докладному звіті.

Є два способи скласти звіт: вибрати шаблон з застосунку або створити свій власний на основі шаблону корпоративної звітності:

– Налаштування змісту.

– Шаблон для повного звіту.

– Список точок доступу й антен.

– Конфігурації точки доступу (ТД).

– Карти покриття й продуктивності.

– Вимоги до мережі.

– Стан мережі й проблеми.

– Окремі звіти для 2,4 і 5 ГГц.

– Покриття кожної точки доступу й SNR.

– Вимоги до ємності й вузькі місця.

– Обстежені райони.

– Зауваження по установці устаткування.

– Докладні показники продуктивності системи визначення місця розташування в реальному часі (RTLS).

– Формати Microsoft Word і PDF.

Покриття й завантаженість Wi-Fi 6:

Аналізуй не тільки покриття, але й думай про завантаженість мережі. Diagnosing Network Wi-Fi 6 Auto-Planner проектує мережа відповідно до ваших вимог до ємності, а також показує перевантажені точки доступу й зони надмірних голосових викликів VoIP:

– Укажіть зони покриття мережі.

– Визначите своїх клієнтів Wi-Fi 6.

– Визначите використовувані додатки.

– Підготуйтеся до пік мережного трафіку.

– Що настроюються опитування пропускної здатності.

– Рівномірний розподілу трафіку .

– Проектування мережі на основі ємності.

– Імітація впливу нового обладнання Wi-Fi 6.

– Відображення проблем із завантаженістю.

– Теплові карти пропускної здатності.

Підтримка 802.11ax (Wi-Fi 6):

802.11ax збільшує не тільки пропускну здатність Wi-Fi 6 але і його складність. Diagnosing Network Wi-Fi 6 допоможе вам зрозуміти й оптимізувати розгортання мережі на основі давно існуючих і найсучасніших стандартів бездротових мереж Wi-Fi 6.

– Проектування мереж 802.11a / b / g / n / ac / ax.

– Планування каналу для змішаних середовищ 802.11a / b / g / n / ac / ax.

– 2500+ корпоративних точок доступу й антен.

– Пасивні й активні обстеження 802.11ax.

– Аналіз пропускної здатності 802.11ax.

– Аналіз шуму й перешкод 802.11ax.

– Спектральний аналіз (потрібно Sidekick).

– Теплові карти пропускної здатності каналу.

Боротьба з перешкодами

Wi-Fi 6 працює на суспільних частотах, що робить його вразливим для перешкод від інших пристроїв, що використовують той же спектр. Перевіряйте доступність вільного радіочастотного спектра, вимірюючи використання каналу й виявляючи перешкоди, не пов'язані з Wi-Fi 6, від таких джерел, як камери спостереження, системи бездротового зв'язку, гарнітури й мікрофони.

Diagnosing Network Wi-Fi 6 надає різні інструменти для усунення неполадок і мінімізації проблем з перешкодами під час обстеження Wi-Fi 6, включаючи автоматичне виявлення перешкод.

Це дозволяє користувачам легко визначити й знайти точне джерело, що викликає проблеми з Wi-Fi 6:

– Візуалізація перешкод і рівнів відносини сигнал-шум SNR на плані будинку.

– Відображення й мінімізація перешкод у сполученому каналі.

– Аналіз використання спектра.

– Обробка результатів на місці й пост фактум.

– Аналіз спектра, відображуваний у вигляді амплітуди залежно від щільності частоти, а також у часі

– Повне відображення спектра й поведінки Wi-Fi 6 у діапазонах 2,4 і 5 ГГц.

Сумісність із операційними системами

– Diagnosing Network Wi-Fi 6 споконвічно працює на MacOS і Windows.

3.4 Розробка діаграми процесів

Відповідно до методичних рекомендацій розроблення графічної частини кваліфікаційної бакалаврської роботи розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3.

Розроблена діаграма взаємодії процесів використовується для представлення та візуалізації процесів обробки даних тобто структурного проектування бакалаврської роботи.

Основні складові елементи діаграми взаємодії процесів це потоки даних:

– Репозиторії, потік сховища даних.

– Потоки зовнішні по відношенню до системи сутності.

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Потоки даних гібридні між елементами трьох попередніх типів.

Відповідно до документації основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей.

Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників.

Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

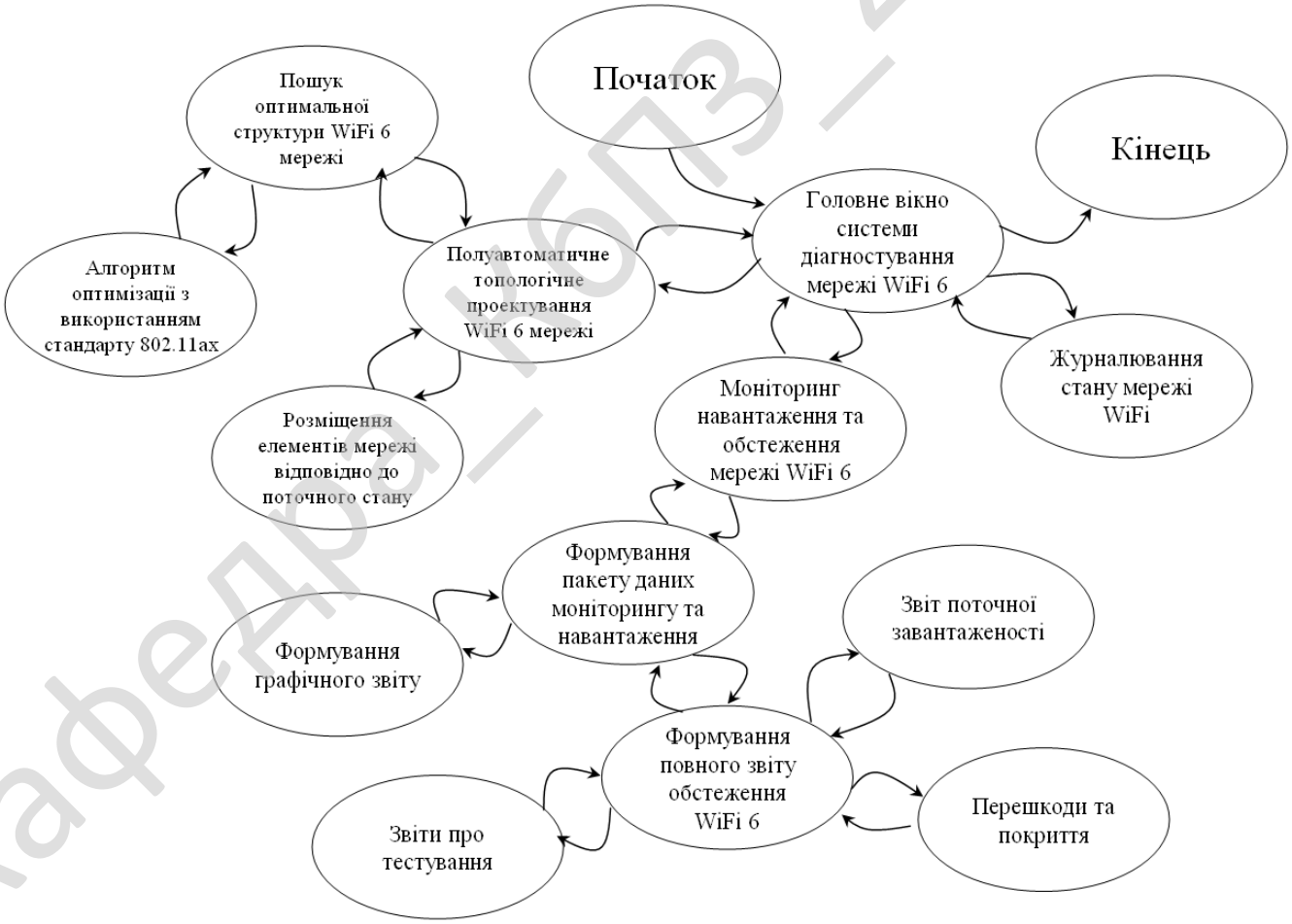

Рисунок 3.3 – Діаграма взаємодії процесів

Для схематичного представлення системи що розробляється необхідно спочатку представити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи в цілому у подальшому. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Розроблена діаграма взаємодії процесів системи в подальшому уточнюється шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Таким чином у результаті після розгляду, вищеописаної системи, схеми структурної, функціональної, діаграми взаємодії процесів перейдемо до опису та розгляду блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми. Його блок-схема зображена на рисунку 4.1, з рисунку видно, що після запуску програми відбуваються наступні дії:

– Виведення вікна діагностування мережі WiFi 6.

– Запит топологічного проектування WiFi 6.

– Введення списку знайденого обладнання (802.11ax).

– Виклик підпрограми планування покриття WiFi 6.

– Виведення результатів роботи підпрограми.

– Запит пакету навантаження.

– Сканування мережі WiFi 6.

– Формування пакету даних.

– Виведення результатів сканування.

– Запит аналізу даних.

– Аналіз статистики навантаження та формування звітів.

– Виведення результатів аналізу.

– Запис результату до журналу роботи ПЗ.

– Формування звіту сумісності з ОС.

– Кінцевий запит завершення роботи ПЗ.

Робота підпрограми зображено на рисунку 4.2, з рисунку видно, що після виклику відбуваються наступні дії:

– Початкове формування таблиць пакету даних.

– Запит планування.

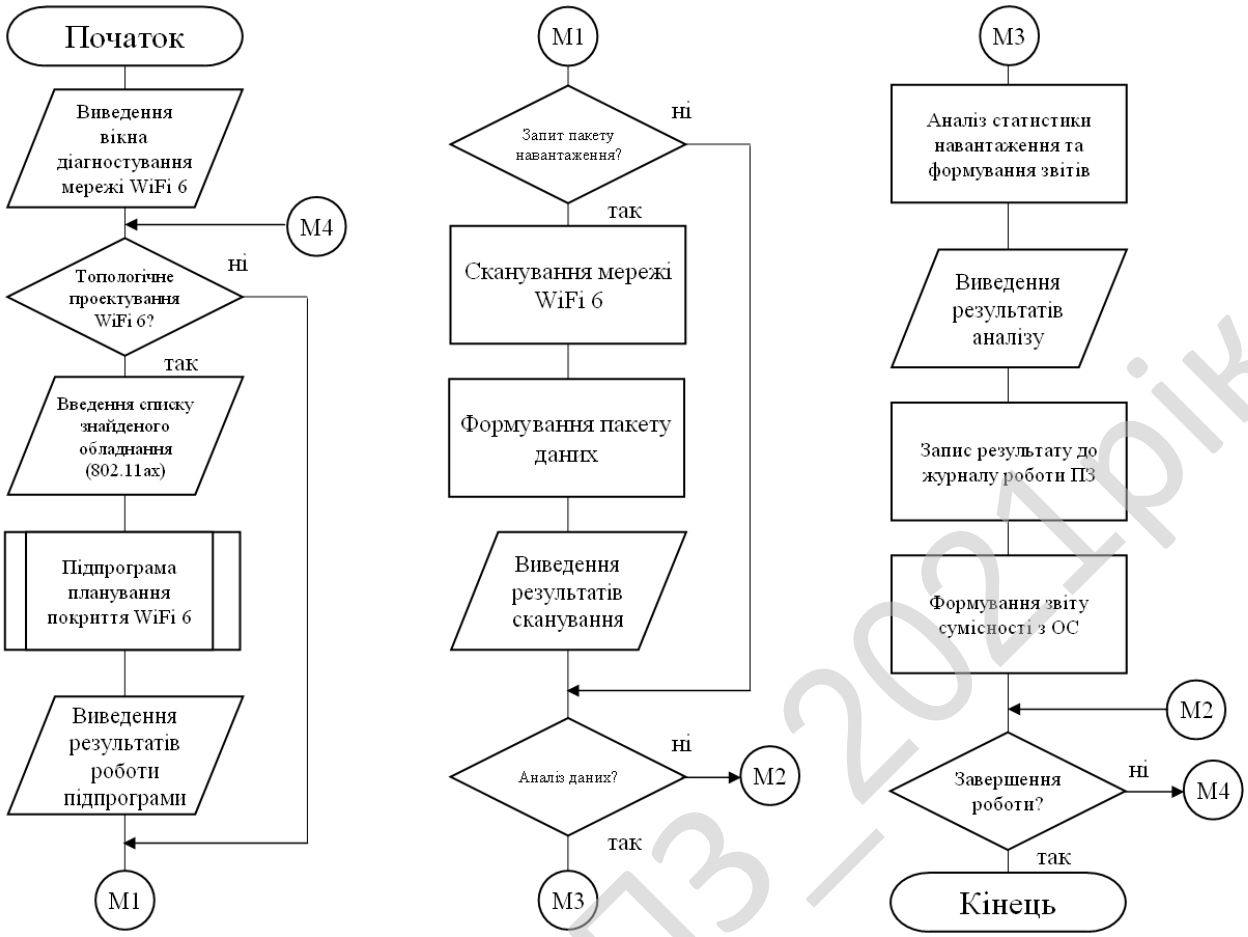

Рисунок 4.1 – Блок-схема основної програми

– Оновлення таблиці відстаней з використанням стандарту 802.11ax.

– Пошук найкоротших шляхів.

– Побудова таблиці відстаней.

– Оновлення таблиці топології мережі.

– Перевірка кількості рядків в таблиці топології.

– Запит перевірки – кількість рядків в таблиці більша за 1?.

– Формування множини вузлів, найближчих до маршрутизатора.

– Оновлення таблиці топологій.

– Запит є повтори в таблиці топологій.

– Оновлення таблиці відстаней.

– Формування базової множини шляхів передачі даних.

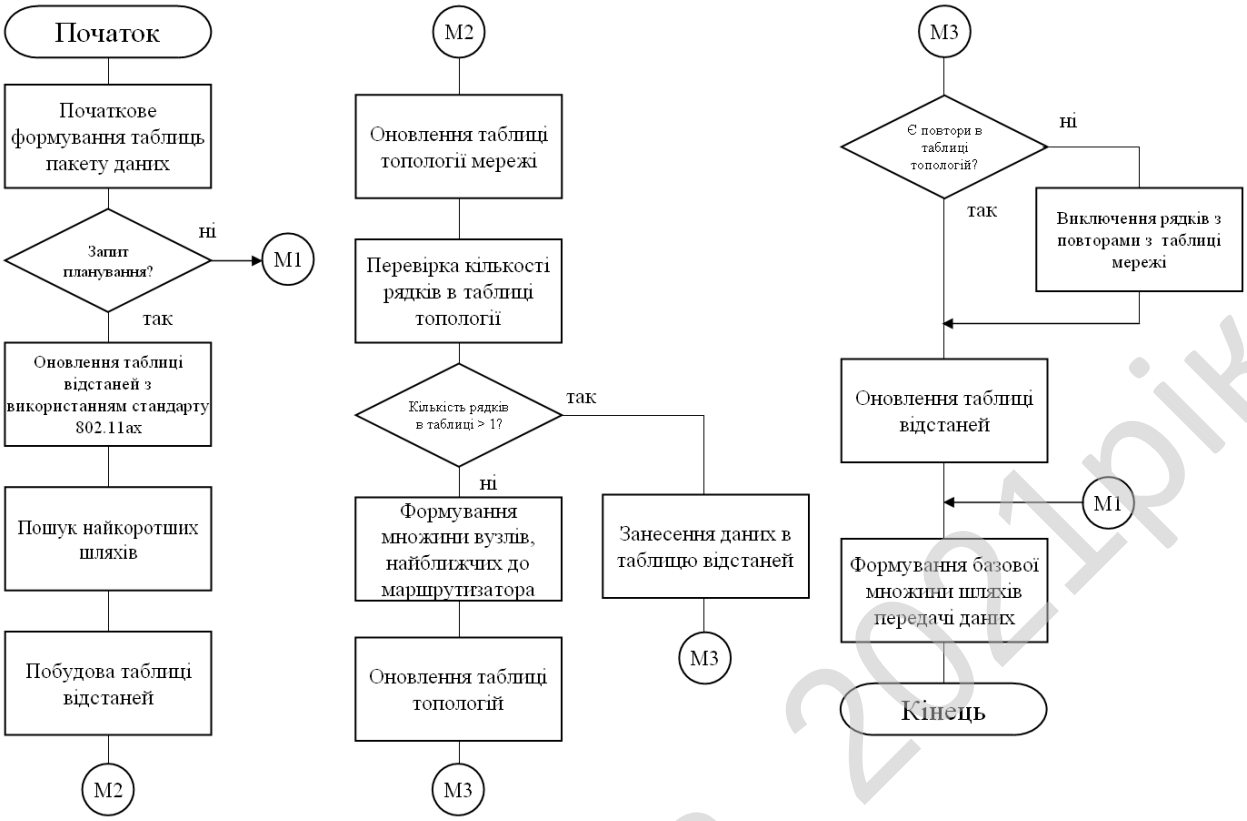

Рисунок 4.2 – Блок-схема роботи підпрограми

Розглянемо реалізацію бакалаврської роботи. Були проведені розрахунки і підібрані набори тестових даних для перевірки правильності реалізації проектних рішень. Розглянемо протокол SNMP (Simple Network Management Protocol, простий протокол керування мережею) – це протокол керування мережами зв'язку на основі архітектури TCP/IP.

На основі концепції TMN в 1980–1990 р. різними органами стандартизації був вироблений ряд протоколів керування мережами передачі даних з різним спектром реалізації функцій TMN. До одного з типів таких протоколів керування належить Simple Network Management Protocol (SNMP).

SNMP – це технологія, покликана забезпечити керування й контроль за пристроями й програмами в мережі зв'язку шляхом обміну керуючою інформацією між агентами, що розташовуються на мережних пристроях, і менеджерами, розташованими на станціях керування. SNMP визначає мережу як

сукупність мережних керуючих станцій й елементів мережі (головні машини, шлюзи й маршрутизатори, термінальні сервери), які спільно забезпечують адміністративні зв'язки між мережними керуючими станціями й мережними агентами. SNMP різних версій присвячений цілий ряд рекомендацій IETF (RFC).

Зазвичай при використанні SNMP присутні керовані та керуючі системи. До складу керованої системи входить компонент, який називається агентом, який відправляє звіти керуючій системі. По суті SNMP агенти передають управлінську інформацію на керуючі системи як змінні (такі як «вільна пам'ять», «ім'я системи», «кількість працюючих процесів» тощо).

Керуюча система може отримати достовірну інформацію через операції протоколу GET, GETNEXT і GETBULK. Агент може самостійно без запиту надсилати дані, використовуючи операцію протоколу TRAP або INFORM. Управляючі системи можуть також відправляти конфігураційні оновлення або контролюючі запити, використовуючи операцію SET для безпосереднього управління системою. Операції конфігурування та управління використовуються тільки тоді, коли потрібні зміни у мережній інфраструктурі. Операції моніторингу зазвичай виконуються на регулярній основі.

Змінні, доступні через SNMP, організовані в ієрархії. Ці ієрархії та інші метадані (такі як тип і опис змінної) описуються Базами Керуючої Інформації' Management Information Bases (MIBs).

SNMP не визначає, яку інформацію (які змінні) керована система повинна надавати. Навпаки, SNMP використовує розширювану модель, в якій доступна інформація визначається Базами Керуючої Інформації (MIB – Management Information Base).

Бази Керуючої Інформації описують структуру керуючої інформації пристроїв. Вони використовують ієрархічний адресний простір імен, що містить унікальний ідентифікатор об'єкта (object identifier (OID)).

Грубо кажучи, кожен унікальний ідентифікатор об'єкта ідентифікує змінну, яка може бути прочитана чи встановлена через SNMP. MIB'и використовують нотацію, визначену в ASN.1.

Ієрархія MIB може бути зображена як дерево з безіменним коренем, рівні якого приписані різними організаціями. На найвищому рівні MIB OID'и належать різним організаціям, що займаються стандартизацією, в той час як на нижчих рівнях OID'и виділяються асоційованими організаціями. Ця модель забезпечує управління на всіх шарах мережної моделі OSI, адже MIB'и можуть бути визначені для будь-яких типів даних і операцій.

Керований об'єкт – це одна з будь-якого числа характеристик, специфічних для керованого пристрою. Керований об'єкт включає в себе один або більше екземплярів об'єкта (що ідентіфікуються за OID), які насправді є змінними.

Існує два типи керованих об'єктів: Скалярні об'єкти визначають єдиний екземпляр об'єкта; Табличні об'єкти визначають множинні, пов'язані екземпляри об'єктів які групуються в таблицях MIB.

Прикладом керованого об'єкта може бути atInput, який є скалярним об'єктом що містить єдиний екземпляр об'єкта, ціле число, яке показує загальну кількість вхідних пакетів AppleTalk на мережний інтерфейс маршрутизатора.

Ідентифікатор об'єкта (OID) унікально ідентифікує керований об'єкт в ієрархії MIB.

В телекомунікаціях і комп'ютерних мережах, ASN.1 є стандартною гнучкою нотацією для опису структур даних, що служать для кодування, передачі і декодування даних. ASN.1 являє собою набір правил для опису структури об'єктів, незалежних від специфічних для обладнання методик кодування, і формальну нотацію, яка дозволяє уникнути неоднозначностей.

ASN.1 це єдиний ISO та ITU-T стандарт, спочатку визначений у 1984 році як частина стандарту CCITT X.409: 1984. Пізніше, в 1988 році, завдяки його широкому застосуванню, він був виділений в окремий стандарт X.208. Значно переглянута версія 1995року описана в X.680.

Адаптована підмножина ASN.1 Структура Керуючої Інформації (SMI) описана в протоколі SNMP для визначення наборів пов'язаних MIB об'єктів, також званих MIB модулями.

SNMP працює на прикладному рівні TCP/IP (сьомий рівень моделі OSI). Агент SNMP отримує запити по UDP-порту 161. Менеджер може посилати запити з будь-якого доступного порту джерела на порт агента. Відповідь агента буде відправлений назад на порт джерела на менеджері. Менеджер отримує повідомлення (Traps і InformRequests) по порту 162. Агент може генерувати повідомлення з будь-якого доступного порту. При використанні TLS або DTLS запити виходять по порту 10161, а пастки відправляються на порт 10162.

У SNMPv1 зазначено п'ять основних протокольних одиниць обміну (protocol data units – PDU). Ще дві PDU, GetBulkRequest і InformRequest, були введені в SNMPv2 і перенесені в SNMPv3.

Нижче перераховані сім протокольних одиниць обміну SNMP:

1. GetRequest. Запит від менеджера до об'єкту для отримання значення змінної або списку змінних. Необхідні змінні вказуються в полі variable bindings (розділ поля values при цьому не використовується). Отримання значень зазначеної змінної повинно бути виконано агентом як Атомарна операція. Менеджеру буде повернений Response (відповідь) з поточними значеннями.

2. SetRequest. Запит від менеджера до об'єкту для зміни змінної або списку змінних. Зв'язані змінні вказуються в тілі запиту. Зміни всіх зазначених змінних повинні бути виконані агентом як атомарна операція. Менеджеру буде повернений Response з (поточними) новими значеннями змінних.

3. GetNextRequest. Запит від менеджера до об'єкту для виявлення доступних змінних і їх значень. Менеджеру буде повернений Response зі зв'язаними змінними для змінної, яка є наступною в базі MIB в лексіграфіческом порядку. Обхід всієї бази MIB агента може бути проведений ітераційним використанням GetNextRequest, починаючи з OID 0. Рядки таблиці можуть бути прочитані, якщо вказати в запиті OID-и колонок в пов'язаних змінних.

4. GetBulkRequest. Покращена версія GetNextRequest. Запит від менеджера до об'єкту для численних ітерацій GetNextRequest. Менеджеру буде повернений Response з декількома пов'язаними змінними, обійденими починаючи зі пов'язаної змінної (змінних) в запиті. Специфічні для PDU поля non-repeaters і max-repetitions використовуються для контролю за поведінкою відповіді. GetBulkRequest був введений в SNMPv2.

5. Response. Повертає зв'язані змінні і значення від агента менеджеру для GetRequest, SetRequest, GetNextRequest, GetBulkRequest і InformRequest. Повідомлення про помилки забезпечуються полями статусу помилки і індексу помилки.

Ця одиниця використовується як відповідь і на Get-, і на Set -запити, в SNMPv1 називається GetResponse.

6. Trap. Асинхронне повідомлення від агента – менеджеру. Включає в себе поточне значення sysUpTime, OID, що визначає тип trap (пастки), і необов'язкові зв'язані змінні. Адресація одержувача для пасток визначається за допомогою змінних trap-конфігурації в базі MIB. Формат trap-повідомлення був змінений в SNMPv2 і PDU перейменували в SNMPv2-Trap.

7. InformRequest. Асинхронне повідомлення від менеджера менеджеру або від агента менеджеру. Повідомлення від менеджера менеджеру були можливі вже в SNMPv1 (за допомогою Trap), але SNMP зазвичай працює на протоколі UDP, в якому доставка повідомлень не гарантована, і не повідомляється про втрачені пакетах. InformRequest виправляє це зворотним відправленням підтвердження про отримання. Одержувач відповідає Response-му, що повторює всю інформацію з InformRequest. Цей PDU був введений в SNMPv2.

Наведемо частину коду, який реалізує вище перераховані дії.

```
procedure TForm1.ClickMeClick(Sender: TObject);
var
 N: TNetworkNeighborhood;
 List: TStringList;
 i, j: Integer;
 ListViewItem: TListItem;
```

```
        WorkgroupNode, ComputerNode: TTreeNode;
      begin
       Screen.Cursor:=crHourGlass;
       try
      // Ініціалізація об’єктів та сканування мережі
        N:=TNetworkNeighborhood.Create;
        try
      // Отримання списку всіх комп’ютерів в мережі
         Memo1.Lines.Clear;
         N.ListComputers(Memo1.Lines);
         // Отримання списку всіх робочих груп і комп’ютерів в мережі,
відсортованих в алфавітному порядку
         List:=TStringList.Create;
         ListView1.Items.Clear;
         try
          N.ListNetwork(List);
          for i:=0 to List.Count - 1 do begin
           ListViewItem:=ListView1.Items.Add;
           ListViewItem.Caption:=List[i];
           ListViewItem.ImageIndex:=Integer(List.Objects[i]);
          end;
         finally
          List.Free;
         end;
         // Побудова дерева робочих груп і комп’ютерів в мережі
         TreeView1.Items.Clear;
         for i:=0 to N.Count - 1 do begin
          WorkgroupNode:=TreeView1.Items.Add(nil, N[i]);
          WorkgroupNode.ImageIndex:=1;
          WorkgroupNode.SelectedIndex:=1;
          for j:=0 to (N.Objects[i] as TStrings).Count - 1 do begin
           ComputerNode:=TreeView1.Items.AddChild(WorkgroupNode, (N.Objects[i] as
TStrings).Strings[j]);
           ComputerNode.ImageIndex:=0;
          end;
         end;
        // Отримання IP адрес комп’ютерів
        GetIPAddresses(N, Memo2.Lines);
        finally
         N.Free;
        end;
        TreeView1.FullExpand;
```

```
 finally
  Screen.Cursor:=crDefault;
 end;
end;
```

Визначимо типи даних та константи, які потрібні для роботи програмного продукту.

```
const
WellKnownPorts: array[1..32] of TWellKnownPort  = (
( Prt: 0; Srv:   ' RESRVED'  ),     {Зарезервовано}
( Prt: 7; Srv:   ' ECHO   '  ),     {Пінгування    }
( Prt: 9; Srv:   ' DISCARD'  ),
( Prt: 13; Srv:  ' DAYTIME'  ),
( Prt: 17; Srv:  ' QOTD   '  ),     {Показчик на день}
( Prt: 19; Srv:  ' CHARGEN'  ),  {Генератор символів}
( Prt: 20; Srv:  ' FTPDATA'  ),  { Протокол FTP – дані}
( Prt: 21; Srv:  ' FTPCTRL'  ),  { Протокол FTP – управління}
( Prt: 22; Srv:  ' SSH    '  ),
( Prt: 23; Srv:  ' TELNET '  ),
(Prt:25;Srv: ' SMTP   '  ),  { Протокол Simple Mail Transfer Protocol}
( Prt: 37; Srv:  ' TIME   '  ),     { Протокол Time Protocol }
( Prt: 43; Srv:  ' WHOIS  '  ),     { WHO IS сервіс  }
( Prt: 53; Srv:  ' DNS    '  ),     { Domain Name Service }
( Prt: 67; Srv:  ' BOOTPS '  ),     { BOOTP сервер }
( Prt: 68; Srv:  ' BOOTPC '  ),     { BOOTP клієнт }
( Prt: 69; Srv:  ' TFTP   '  ),     { Стандартний  FTP  }
( Prt: 70; Srv:  ' GOPHER '  ),     { Протокол Gopher       }
( Prt: 79; Srv:  ' FINGER '  ),     { Протокол Finger       }
( Prt: 80; Srv:  ' HTTP   '  ),     { Протокол HTTP         }
( Prt: 88; Srv:  ' KERBROS'  ),     { Протокол Kerberos     }
( Prt: 109; Srv: ' POP2   '  ),{ Протокол Post Office Protocol Version 2 }
( Prt: 110; Srv: ' POP3   '  ),{ Протокол Post Office Protocol Version 3 }
( Prt: 111; Srv: ' SUN_RPC'  ),     { SUN віддалений виклик функцій }
( Prt: 119; Srv: ' NNTP   '  ),{ Протокол Network News Transfer Protocol }
( Prt: 123; Srv: ' NTP    '  ),     { Протокол Network Time protocol}
( Prt: 135; Srv: ' DCOMRPC'  ),     { Локальний сервіce             }
( Prt: 137; Srv: ' NBNAME '  ),     { NETBIOS сервіс імен          }
( Prt: 138; Srv: ' NBDGRAM'  ),     { NETBIOS сервіс датаграм     }
( Prt: 139; Srv: ' NBSESS '  ),     { NETBIOS сервіс сесій        }
( Prt: 143; Srv: ' IMAP   '),{ Протокол Internet Message Access Protocol }
( Prt: 161; Srv: ' SNMP   '),{ Протокол Simple Netw. Management Protocol }
( Prt: 169; Srv: ' SEND   '  ));
```

```
const
  ICMP_ERROR_BASE = 11000;
type
TNetworkParams = record
    HostName: string ;
    DomainName: string ;
    CurrentDnsServer: string ;
    DnsServerTot: integer ;
    DnsServerNames: array [0..9] of string ;
    NodeType: UINT;
    ScopeID: string ;
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;
TIfRows = array of TMibIfRow ; // динамічний масив колонок
```

Jira – була використана комерційна система відслідковування помилок, призначена для організації взаємодії з користувачами, хоча в деяких випадках використовується і для управління проектами. Розроблено компанією Atlassian, є одним з двох її основних продуктів (поряд з вікі-системою Confluence). Має веб-інтерфейс.

Назва системи отримано шляхом усічення слова «Gojira» - Японського імені монстра Годзилла, що, в свою чергу, є відсиланням до назви конкуруючого продукту - Bugzilla; створювалася в якості заміни Bugzilla і багато в чому повторює її архітектуру. Система дозволяє працювати з декількома проектами. Для кожного з проектів створює і веде схеми безпеки і схеми оповіщення.

До версії 3.13.5 (включно) розрізнялися редакції Enterprise, Professional і Standard, після - Залишилася тільки редакція Enterprise (для великих організацій).

Система заснована на Java EE і працює на кількох популярних системах управління базами даних і операційних системах.

Основний елемент обліку в системі - завдання (ticket або issue). Завдання містить назву проекту, тему, тип, пріоритет, компоненти і зміст. Завдання може бути розширена додатковими полями (також і нові призначені для користувача поля можуть бути визначені), додатками (наприклад - Фотографіями,

скриншотами) або коментарями. Завдання може редагуватися або просто змінювати статус, наприклад, з «відкритий» в «закритий». Які переходи між станами можливі, визначається через настроюється потік операцій. Будь-які зміни в задачі записуються в журнал.

Jira має велику кількість можливостей конфігурації: для кожної програми може бути визначений окремий тип завдання з власним workflow, набором статусів, одним або декількома видами уявлення (screens). Крім того, за допомогою так званих «схем» можна визначити для кожного індивідуального Jira-проекту власні права доступу, поведінку і видимість полів і багато іншого.

Завдяки універсальному підходу можна пристосувати Jira для багатьох непрофільних завдань, наприклад, керування вимогами, керування ризиками, аж до реалізації невеликої системи бронювання, автоматизації процесу рекрутингу.

Для інтеграції з зовнішніми системами підтримує інтерфейси SOAP, XML-RPC і REST. Поставляється із засобами інтеграції з такими системами управління версіями, як Subversion, CVS, Git, Clearcase, Team Foundation Server, Mercurial і Perforce.

Існують доповнення, що дозволяють вбудувати Jira в інтегровані середовища розробки, в тому числі Eclipse і IntelliJ IDEA. Перекладена багатьма мовами, включаючи російську, англійську, японську, німецьку, французьку, іспанську.

Для сторонніх розробників надаються кошти розробки розширень системи - плагінів. Розробники розширень можуть викладати плагіни для продажу на спеціальний розділ сайту Atlassian.

Є комерційним продуктом, який може бути ліцензований для роботи на локальному сервері або доступний в якості віддаленого додатки. Ціноутворення залежить від максимального числа користувачів, при цьому близько $50 за користувача для локального і $7 на місяць за користувача для віддаленого доступу є типовими цінами.

Для академічних і комерційних клієнтів доступний повний вихідний код під ліцензією розробника.

Для проектів з відкритим вихідним кодом Atlassian надає спеціальну безкоштовну ліцензію при дотриманні наступних правил:

– проект використовує ліцензії, схвалені Open Source Initiative;

– Вихідний код проекту доступний для скачування;

– у проекту є публічно доступна веб-сайт;

– програмне забезпечення від Atlassian є на веб-сайті проекту.

Була використана водоспадна (каскадна) модель життєвого циклу ПЗ (waterfall model) – послідовний метод розробки програмного забезпечення, названий так через діаграму схожу на водоспад.

Ця модель розробки запозичена з системної інженерії у виробництві та будівництві - областях, в яких зміни на пізніх етапах дуже дорогі, або неможливі. Наприклад, для створення складних інженерних конструкцій (споруд, літаків, мостів і т.п.). Зміни в проекті фундаменту будинку після того, як покладений дах коштують дуже дорого, тому перфекціонізм на початкових етапах проектування просто необхідний. Інженери, які починали займатись розробкою програмного забезпечення перейшовши з інших галузей, просто адаптували звичну модель, тому що на ранніх етапах розвитку комп'ютерної техніки не було методологій створених саме для програмування. Проте, схожі методології застосовуються для програмного забезпечення й далі, у випадках коли вимоги фіксовані, і вимагається висока якість та надійність, наприклад в системах для військових чи медичних потреб.

Перший формальний опис водоспадної моделі, після якої вона стала популярною був здійснений В. В. Ройсом у 1970. Попри те, що стаття містить переважно критику методу, на неї часто посилаються.

Переваги методу:

– Ніяких переробок.

– Гарна специфікація перетікає в гарну документацію.

– Зрозуміла модель.

– Розробники можуть мати низьку кваліфікацію.

Недоліки:

– Необхідний перфекціонізм на кожному етапі.

– Важко вносити зміни (якщо взагалі можливо).

– Надлишкове проектування.

– Поділ розробників на "perfect" та "code monkeys".

Модифікації. Через те що цей метод погано підходить для розробки саме ПЗ, частіше використовують його модифікації.

Найвідоміша модифікація - Sashimi. Названа так через японську страву сашимі (суші нарізане і сервіроване так, що складені рядочком шматочки накладаються один на одного). В моделі розробки Сашимі фази життєвого циклу йдуть одна за одною, але при цьому перекриваються одна з одною в часі.

Розглянемо визначення API. Це прикладни́й програ́мний інтерфе́йс (Application Programming Interface, API) – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення.

Спрощено - це набір чітко визначених методів для взаємодії різних компонентів. API надає розробнику засоби для швидкої розробки програмного забезпечення. API може бути для веб-базованих систем, операційних систем, баз даних, апаратного забезпечення, програмних бібліотек.

Одним з найпоширеніших призначень API є надання набору широко використовуваних функцій, наприклад для малювання вікна чи іконок на екрані. API є абстрактним поняттям – програмне забезпечення, що пропонує деякий API, часто називають реалізацією (implementation) даного API. У багатьох випадках API є частиною набору розробки програмного забезпечення, водночас, набір розробки може включати як API, так і інші інструменти/апаратне забезпечення, отже ці два терміни не є взаємозамінювані.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм LOKI_91. Механізм алгоритму LOKI_91 подібний DES (рисунок 4.3).

Блок даних розщеплюється на ліву й праву половини й проходить 16 раундів, що досить нагадує DES.

У кожному раунді права половина спочатку піддається операції XOR із частиною ключа, а потім розширювальній перестановці (таблиця 4.1).

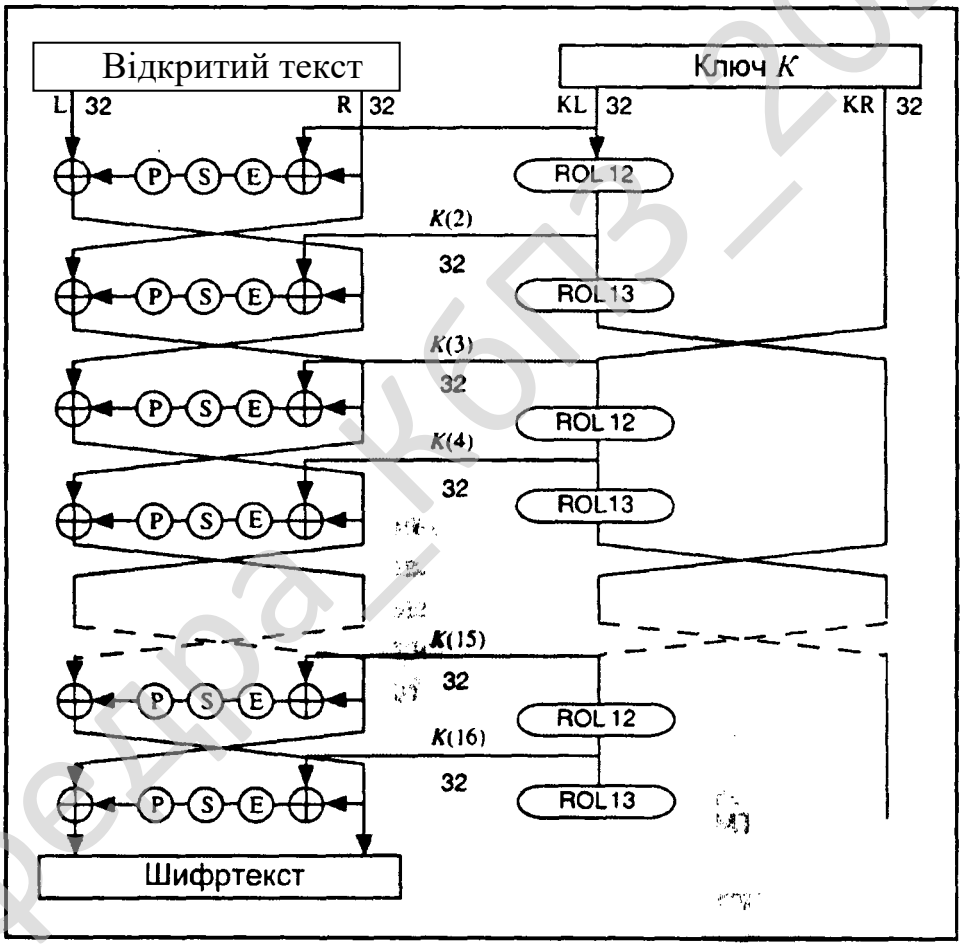

Рисунок 4.3 – Алгоритм LOKI91

Таблиця 4.1 – Перестановка з розширенням

4	3	2	1	32	31	30	29	28	27	26	25
28	27	26	25	24	23	22	21	20	19	18	17
20	19	18	17	16	15	14	13	12	11	10	9
12	11	10	9	8	7	6	5	4	3	2	1

48-бітовий вихід розділяється на чотири 12-бітових блоки. У кожному блоці виконується така підстановка з використанням S-блоку: береться кожний 12-бітовий вхід, 2 старших і 2 молодших біти використовуються для утворення номера r, а вісім внутрішніх біт утворять номер c. Вихід S-блоку, O, має наступне значення: $O(r,c) = (c + ((r*17) \oplus 0xff) \ \& \ 0xff)^{31} \bmod P_r$.

Таблиця 4.2 – Значення P_r

r	1	2	3	4	5	6	7	8
P_r	375	379	391	395	397	415	419	425
r	9	10	11	12	13	14	15	16
P_r	433	445	451	463	471	477	487	499

Після цього чотири 8-бітових результати знову поєднуються, утворюючи 32-бітове число, що піддається операції перестановки, описаній в таблиці 3. Нарешті, для одержання нової лівої половини виконується операція XOR правої половини з колишньою лівою половиною, а ліва половина стає новою правою половиною. Після 16 раундів для одержання остаточного шифртексту знову виконується операція XOR над блоком і ключем.

Таблиця 4.3 – Перестановка за допомогою P-блоку

32	24	16	8	31	23	15	7	30	22	14	6	29	21	13	5
2	20	12	4	27	19	11	3	26	18	10	2	25	17	9	1

Підключи генеруються із ключа досить прямолінійно. 64-бітовий ключ розбивається на ліву й праву половини.

На кожному раунді підключем служить ліва половина. Далі вона циклічно зрушується вліво на 12 або 13 біт, потім після кожних двох раундів ліва й права половини міняються місцями.

Як і в DES, для зашифрування й розшифрування використовується один й той самий алгоритм із деякими змінами у використанні підключів.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1. З рисунку головного вікна можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Функціональних кнопок ПЗ: Сканування; Статистика; Налаштування фільтру.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Верхнього меню: Формат; Дані; Налаштування; Довідкова.

– Розділу обрання групи: Підключення; Поточний стан системи.

– Розділу виведення результату роботи системи: Інфо.

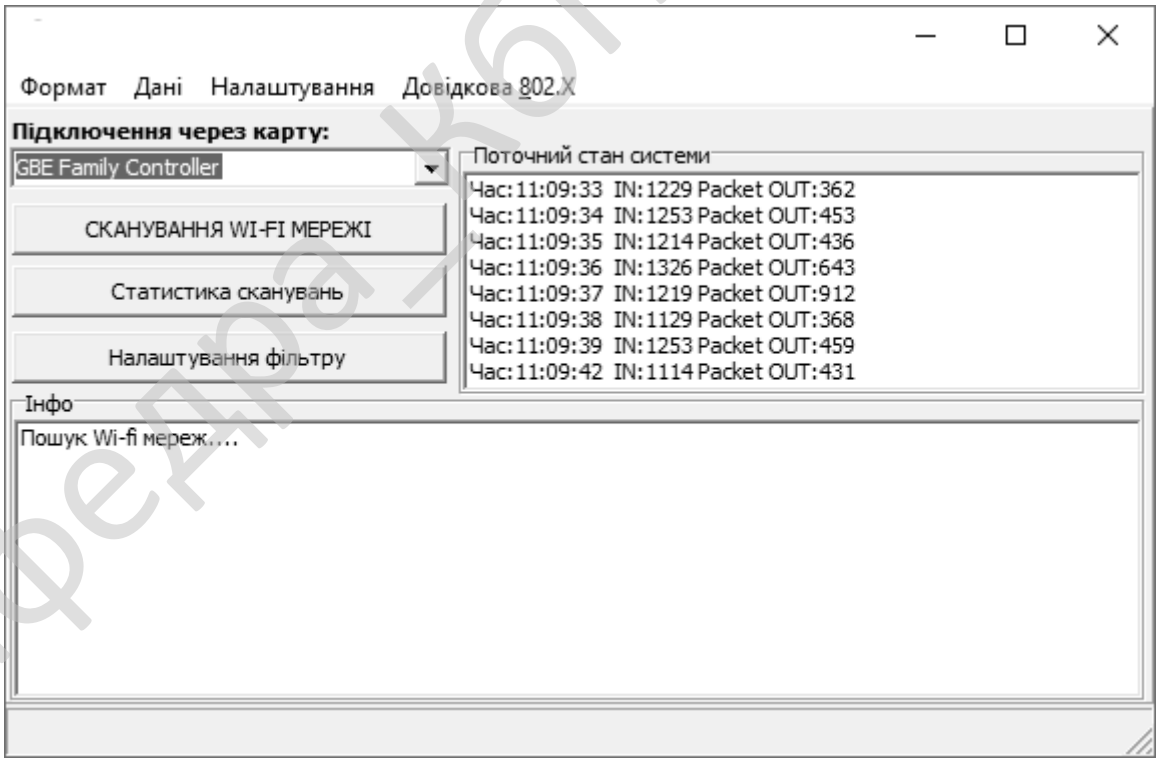

Рисунок 5.1 – Головне вікно ПЗ

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

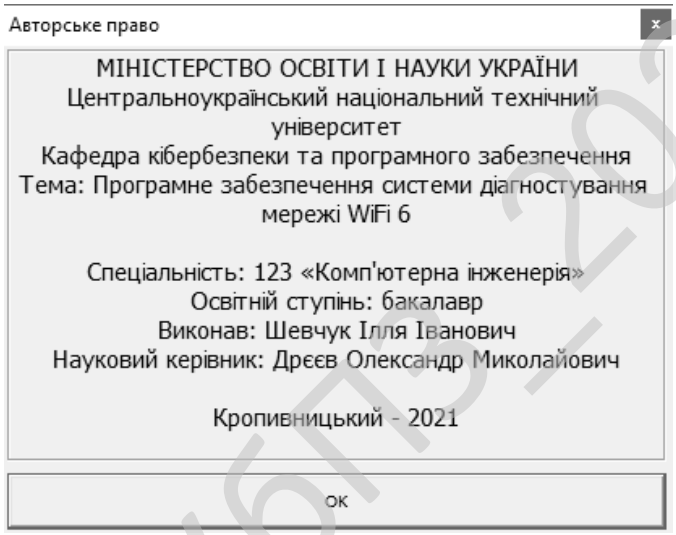

Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання кваліфікаційної бакалаврської роботи, призначено для системи діагностування мережі Wi-Fi 6.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем діагностування мережі Wi-Fi 6.

– Досліджена система діагностування мережі Wi-Fi 6.

– На основі отриманих результатів досліджень створена програмна реалізація системи діагностування мережі Wi-Fi 6.

Розроблені під час виконання кваліфікаційної бакалаврської роботи алгоритми дозволяють успішно вирішувати завдання діагностування мережі Wi-Fi 6.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня RAD Studio Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи діагностування мережі Wi-Fi 6. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм LOKI_91.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Майника Э. Алгоритмы оптимизации на сетях и графах: пер. с англ. / Э. Майника; под ред. Е.К. Масловского. – М.: Мир, 1981. – 321 с.

2. МСЭ-Т Рекомендация G.101. Международные телефонные соединения и цепи – Общие определения //11/2003. [Електронний ресурс]. – Режим доступу до ресурсу: http://www.telecom61.ru/SharedFiles/ Download.aspx? …pageid=106

3. Назаров А.Н. Модели и методы расчета структурно-сетевых параметров АТМ сетей / Алексей Николаевич Назаров. – М.: Горячая линия - Телеком, 2002. – 256 с.

4. Одом Ш. Коммутаторы CISCO / Ш. Одом, Х. Ноттингем – М.: "Кудиц-Образ", 2003. – 528 с.

5. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов / В.Г. Олифер, Н.А. Олифер. – 2-е изд. – СПб.: Питер, 2007. – 958 с.

6. Партыка С.А. Метод ускоренной коррекции SPT с использованием динамических алгоритмов [Електронний ресурс]. – Режим доступу до ресурсу: http://openarchive.nure.ua/bitstream/123456789/936/1/ ASU_158_2012%20%2842-47%29.pdf

7. Руководство по технологиям объединенных сетей. 4-е изд. / пер.с англ. и ред. А.Н. Крикуна – М.: Изд. дом «Вильямс», 2005. – 1040 с.

8. Семенов С.Г. Математическая модель мультисервисного канала связи на основе экспоненциальной GERT-сети / С.Г. Семенов, Є.В. Мелешко, Я.В. Ілюшко // Системи озброєння і військова техніка. – Х.:ХУ ПС. – 2011. –Вип. 3(27). – С. 64-67.

9. Семенов С.Г. Математична модель системи криптографічного захисту електронних повідомлень на основі GERT-мережі / С.Г. Семенов,

О.О. Сур // Системи управління, навігації та зв'язку. – К.: ЦНДІ навігації і управління. – 2012. – Том 1. Вип. 1(21). – С. 131-137

10. Семенов С.Г. Исследования вероятностно-временных характеристик мультисервисного канала связи с использованием математического аппарата GERT-сети / С.Г. Семенов, В.В. Босько, І.А. Березюк // Системи обробки інформації. – Х.: ХУ ПС, 2012. – Т. 1., Вип. 3(101). – С. 139-142.

11. Семенов С.Г. Моделирование защищенного канала связи с использованием экспоненциальной GERT-сети / С.Г. Семенов, А.А. Можаев // Информатика, математическое моделирование, экономика. – Смоленськ.: Смоленский филиал АНО ВПО ЦС РФ "Российский университет кооперации". – 2012. – Том.1. – С. 152-160.

12. Семенов С.Г. Методика математического моделирования защищенной ИТС на основе многослойной GERT-сети / С.Г. Семенов // Вісник Національного технічного університету «Харківський політехнічний інститут». – Х.:НТУ «ХПІ». – 2012. –№62 (968). – С 173-181.

13. Семенов С.Г. Защита данных в компьютеризированных управляющих системах / С.Г. Семенов, В.В. Давыдов, С.Ю. Гавриленко. – LAP Lambert Academic Publishing GmbH & Co. KG (Саарбрюккен, Германия), 2014. – 236 с.

14. Смирнов А.А. Анализ и сравнительное исследование перспективных направлений развития цифровых телекоммуникационных систем и сетей / А.А.Смирнов, В.В.Босько, Е.В.Мелешко // Системи обробки інформації. – Х.: ХУ ПС, 2008. – Вип.7(74). – С.120-123.

15. Смирнов А.А. Усовершенствование метода управления очередями в многопротокольных узлах телекоммуникационной сети / А.А.Смирнов, Е.В.Мелешко // Збірник тез та доповідей другої всеукраїнської науково-практичної конференції «Системний аналіз. Інформатика. Управління». Запоріжжя. Тези доповідей. Запоріжжя: КПУ, 2011.

16. Смирнов С.А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А.К. Дидык, С.А.

Смирнов // Информационные технологии в управлении, образовании, науке и промышленности: монография / Под редакцией профессора В.С. Пономаренко. – Х.: Видавець Рожко С.Г., 2016. – 566 с.

17. Смирнов С. А. Сравнительные исследования математических моделей технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, А. В. Коваленко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2014. – Вип. 9(125). – 105-110.

18. Смирнов С. А. Математическая модель интеллектуального узла коммутации с обслуживанием информационных пакетов различного приоритета / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2014. – Вип. 4 (41). – С. 48-52.

19. Смирнов С. А. Исследование показателей качества функционирования интеллектуальных узлов коммутации в телекоммуникационных системах и сетях / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2014. – № 4(17). – С. 90-95.

20. Смирнов С. А. Усовершенствованный алгоритм управления доступом к «облачным» телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, Н. С. Якименко, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2015. –Вип. 1(126). – С. 150-153.

21. Smirnov S.A. Method of controlling access to intellectual switching nodes of telecommunication networks and systems / A.A. Smirnov, Mohamad Abou Taam, S.A. Smirnov // International Journal of Computational Engineering Research (IJCER). – Volume 5, Issue 5. – India. Delhi. – 2015. – P. 1-7.

22. Смирнов С. А. Анализ и исследование методов управления сетевыми ресурсами для обеспечения антивирусной защиты даннях / Мохамад Абу Таам

Гани, А. А. Смирнов, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2015. – № 3(43). – С. 100-107.

23. Смирнов С. А. Исследование эффективности метода управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Наука і техніка Повітряних Сил Збройних Сил України: наук. журн. –Х.: ХУПС, 2015. – № 3(20). – С. 134-141.

24. Смирнов С. А. Комплекс GERT-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Безпека інформації: наук. - практ. журн. – К.: НАУ, 2015. – Т. 21, № 3. – С. 251-262.

25. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, А. К. Дидык, С. А. Смирнов // Системи озброєння і військова техніка: наук. журн. – Х.: ХУПС, 2016. – № 2 (46). – С. 146-149.

26. Смирнов С. А. Модели системы нейросетевых экспертов безопасной маршрутизации в облачных антивирусных системах / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Системи обробки інформації: зб. наук. праць. – Х.: ХУПС, 2016. – Вип. 3 (140). – С. 36-39.

27. Смирнов С. А. Метод безопасной маршрутизации на базовом множестве путей передачи метаданных в облачные антивирусные системы / В. Л. Бурячок, С. А. Смирнов // Системи управління, навігації та зв'язку. – Полтава, 2016. – Вип. 4(40). – С. 57-62.

28. Смирнов С. А. Способ контроля линий связи телекоммуникационной системы облачного антивируса / А. А. Смирнов, А. К. Дидык, А. Н. Дреев, С. А. Смирнов // Збірник наукових праць Харківського університету Повітряних Сил. – Харків: ХУПС, 2016. – № 2 (47). – С. 148-152.

29. Смирнов С. А. Дослідження та реалізація GERT-моделі технології розповсюдження комп'ютерних вірусів для захисту телекомунікаційних систем /

В. Л. Бурячок, Мохамад Абу Таам Гани, С. А. Смирнов // Інформаційні технології та комп'ютерна інженерія: зб. тез доп. наук.-практ. конф., м. Кіровоград, 4 грудня 2014 р. – Кіровоград: КНТУ, 2014. – С. 168.

30. Смирнов С. А. Исследование математических моделей технологии распространения компьютерных виру сов / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць міжнар. наук.-практ. конф., м. Київ, 25-28 лютого 2015 р. – К.: Європейський університет, 2015. – С. 90-91.

31. Смирнов С. А. Метод управления доступом к «облачным» ресурсам для защиты телекоммуникационных систем / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Всеукраїнська науково-практична конференція «Інформаційна безпека держави, суспільства та особистості», м. Кіровоград, 16 квітня 2015 р.: зб. тез доп. – Кіровоград: КНТУ, 2015. – С. 50-52.

32. Смирнов С. А. Разработка метода управления доступом в интеллектуальных узлах коммутации / А. А. Смирнов, Мохамад Абу Таам Гани, С. А. Смирнов // Проблеми і перспективи розвитку IT-індустрії: зб. тез VII міжнар. наук.-практ. конф., м. Харків, 17-18 квітня 2015 р. – Х.: ХНЕУ, 2015. – С. 14.

33. Смирнов С.А. Реализация метода управления доступом в интеллектуальных узлах коммутации / А.А. Смирнов, Мохамад Абу Таам Гани, С.А. Смирнов // Збірник тез XVII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 17-18 квітня 2015 р. – Кіровоград: КНТУ. – 2015. – С. 91-92.

34. Смирнов С. А. Технология передачи сигнатур в облачные антивирусные системы для обеспечения защищенности телекоммуникационных сетей / А. А. Смирнов, С. А. Смирнов // Збірник тез V міжнародної науково-технічної конференції «ITSEC», Київ, 19-22 травня 2015 р. – К.: НАУ 2015. – С. 12-13.

35. Смирнов С. А. Реализация математической модели интеллектуального узла коммутации для обеспечения защищенности телекоммуникационной сети / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Інформаційна та економічна безпека (INFECO-2015): зб. тез II Міжнар. наук.-практ. Інтернет-конф., м. Харків, 21-22 травня 2015 р. – Х.: ХІБС УБС НБУ, 2015. – С. 20-24.

36. Смирнов С. А. Разработка математической модели технологии распространения компьютерных вирусов в информационно-телекоммуникационных сетях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Сборник тезисов XI международной конференции «Стратегия качества в промышленности и образовании», г. Варна, Болгария, 01-06 июня 2015 г. – Варна: ТУВ, 2015. – С. 488-491.

37. Смирнов С. А. Метод управления доступом к облачным телекоммуни-кационным ресурсам для обеспечения защиты даннях / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Комп'ютерні технології та інформаційна безпека: зб. тез доп. міжнар. наук.-практ. конф., м. Кіровоград, 2-3 липня 2015 р. – Кіровоград: КНТУ, 2015. – С. 4-5.

38. Смирнов С. А. Имитационная модель системы управления доступом к облачным антивирусным телекоммуникационным ресурсам / Мохамад Абу Таам Гани, А. А. Смирнов, С. А. Смирнов // Збірник тез першої всеукраїнської науково-практичної конференції «Перспективні напрями захисту інформації» (м. Затока, 7-9 вересня 2015 р.). – Одеса: ОНАЗ, 2015. – С. 90-94.

39. Смирнов С. А. Разработка комплекса GERT-моделей технологии облачной антивирусной защиты телекоммуникационной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційні технології та взаємодії» (IT & I): зб. тез II міжнар. наук.-практ. конф., м. Київ, 3-5 листопада 2015 р. – К.: КНУ ім. Тараса Шевченка, 2015. – С. 65-67.

40. Смирнов С. А. Разработка моделей телекоммуникационной системы формирования и обработки метаданных в облачных антивирусных системах /

А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Информационные и телекоммуникационные технологии: образование, наука, практика: сб. тезисов II междунар. научно-практ. конф., г. Алматы, Казахстан, 3-4 декабря 2015 г. – Алматы: КазНИТУ им. К.И. Сатпаева, 2015. – С. 309-313.

41. Смирнов С. А. GERT-модели технологии облачной антивирусной защиты / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Безпека українського суспільства в концепції вступу в постіндустріальне суспільство ЄС: зб. тез Круглого столу, м. Київ, 16 грудня 2015 р. – К.: Європейський університет, 2015. – С.41-43.

42. Смирнов С. А. Алгоритмы формирования множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Актуальні питання забезпечення кібернетичної безпеки та захисту інформації: зб. наук. праць II Міжнар. наук.-практ. конф., м. Київ, 24-27 лютого 2016 р. – К.: Європейський університет, 2016. – С. 140-142.

43. Смирнов С. А. Разработка и реализация метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Securitea informationala 2015-2016: Conferenta internationala (editia a XII-a), Chisinau, Moldova, 3 martie 2016. – Chisinau: ADSEM, 2016. – С. 90-96.

44. Смирнов С. А. Алгоритм формирования базового множества маршрутов передачи метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформатика та системні науки (ІСН-2016): зб. тез VII всеукр. наук.-практ. конф., м. Полтава, 10-12 березня 2016 р. – Полтава: ПУЕТ, 2016. – С. 261-263.

45. Смирнов С. А. Система обработки и формирования начального состояния маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми кібербезпеки

інформаційно-телекомунікаційних систем: зб. тез наук.-практ. конф., м. Київ, 10-11 березня 2016 р. – К.: КНУ ім. Тараса Шевченка, 2016. – С. 81-82.

46. Смирнов С. А. Алгоритм безопасной маршрутизации на базовом множестве путей передачи метаданных в программный сервер облачной антивирусной системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна безпека та комп'ютерні технології (IS&CT): зб. тез міжнар. наук.-практ. конф., м. Кіровоград, 24-25 березня 2016 р. – Кіровоград: КНТУ, 2016. – С. 73.

47. Смирнов С. А. Исследование способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Збірник тез першої міжнародної науково-практичної конференції «Проблеми науково-технічного та правового забезпечення кібербезпеки у сучасному світі» (ПНПЗК-2016), м. Харків, 30 березня - 1 квітня 2016 р. – Х.: НТУ «ХПІ», 2016. – С. 14.

48. Смирнов С. А. Разработка способа контроля линий связи телекоммуникационной системы для облачных антивирусов / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Матеріали XVIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування» (м. Кіровоград, 15-16 квітня 2016 р.). –Кіровоград: КНТУ, 2016. – С. 182-186.

49. Смирнов С. А. Разработка и исследование способа контроля линий связи телекоммуникационных сетей для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Проблеми і перспективи розвитку ІТ-індустрії: VIII міжнар. наук.-практ. конф., м. Харків, 28-29 квітня 2016 р.: зб. тез. – Х.: ХНЕУ, 2016. – С. 48.

50. Смирнов С. А. Модель системы нейросетевых экспертов безопасной маршрутизации для облачных антивирусных систем / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Інформаційна та економічна безпека (INFECO-2016): зб. тез III міжнар. наук.-практ. конф., м. Харків, 28-30 кві. 2016 р. – Х.: ХННІ ДВНЗ «УБС», 2016. – С. 178-182.

51. Смирнов С. А. Метод безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. К. Дидык // Сборник тезисов XII международной конференции «Стратегия качества в промышленности и образовании» (г. Варна, Болгария, 30 мая - 02 июня 2016 г.). – Варна: ТУВ, 2016. – С. 581-585.

52. Смирнов С. А. Оценка эффективности метода безопасной маршрутизации метаданных в облачные антивирусные системы / А. А. Смирнов, С. А. Смирнов, А. С. Коваленко // РадіоЕлектроніка та ІнфоКомунікації: зб. тез першої наук. - техн. конф., м. Київ, 11-16 вересня 2016 р. – К.: НТУУ «КПІ», 2016. – С. 17.

53. Современные телекоммуникации. Технологии и экономика / [В.Л. Банкет, О.В. Бондаренко, П.П. Воробьенко и др.]; под ред. С.А. Довгого. – М.: Эко-Трендз, 2003. – 320 с.

54. Столлингс В. Современные компьютерные сети / Вильям Столлингс. – СПб.: Питер, 2003. – 778 с.

55. Таненбаум Э. Компьютерные сети / Эндрю Таненбаум; пер. с англ. А. Леонтьев. – СПб.: Питер, 2002. – 848 с.

56. Телекоммуникационные системы и сети: учебное пособие. В 3 томах / [В.В. Величко, Е.А. Субботин, В.П. Шувалов, А.Ф. Ярославцев]; под ред. В.П. Шувалова. – М.: Горячая линия-Телеком, 2005, т. 3 – 592 с.

57. Хайкин С. Нейронные сети: полный курс / С. Хайкин. – М.:Вильямс, 2006. – 1103 с.

58. Шелухин О.И. Фрактальные процессы в телекоммуникациях: моногр. / О.И. Шелухин, А.М. Тенякшев, А.В. Осин – М.: Радиотехника, 2003. – 480 с.

59. Elwalid, D. Mitra, I. Saniee, and I. Widjaja. Routing and Protection in GMPLS Networks: From Shortest Paths to Optimized Designs // Journal of lightwave technology. – 2003. – №21(11), P. 2828-28-38.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи діагностування мережі WiFi 6.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 204-02 від 28.12.2020 року).

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення системи діагностування мережі WiFi 6.

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи діагностування мережі WiFi 6;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище RAD Studio Delphi 10.4.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 72 аркуші.

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист 22.05.2021 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист 5.06.2021 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник кваліфікаційної бакалаврської роботи

_____ Дрєєв О.М.

Програмне забезпечення системи діагностування мережі WiFi 6

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 39

Літера: РП

Кропивницький – 2021 року

Основна програма

Файл Main.pas основної програми

```
unit Main;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Dim, Networks, StdCtrls, ComCtrls, ImgList, ShellAPI, ShlObj, IpHlpApi,
IPtraff,
  ExtCtrls, About, Buttons;

type
  TForm1 = class(TForm)
    Memo1: TMemo;
    ClickMe: TButton;
    TreeView1: TTreeView;
    ImageList1: TImageList;
    ListView1: TListView;
    Memo2: TMemo;
    Button1: TButton;
    Label1: TLabel;
    Edit1: TEdit;
    Memo3: TMemo;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    MemoTR: TMemo;
    Timer1: TTimer;
    Label7: TLabel;
    Button2: TButton;
    Button3: TButton;
    SpeedButton1: TSpeedButton;
    SpeedButton2: TSpeedButton;
    SpeedButton3: TSpeedButton;
    SpeedButton4: TSpeedButton;
    SpeedButton5: TSpeedButton;
    SpeedButton6: TSpeedButton;
    MemoX: TMemo;
    Label8: TLabel;
    procedure ClickMeClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure SpeedButton2Click(Sender: TObject);
    procedure SpeedButton4Click(Sender: TObject);

  private
    { Private declarations }
     procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
```

```
{$R *.DFM}

function GetShellFolder(CompName: TString): IShellFolder;
var
 S: TString;
 W: WideString;
 P: PWideChar;
 Desktop: IShellFolder;
 Len, Flags: LongWord;
 Machine: PItemIDList;
begin
 S:=CompName;
 if Pos(‘ \\’ , S) <> 1 then S:=‘ \\’ +S;
 Len:=Length(S);
 W:=S;
 P:=@W[1];
 SHGetDesktopFolder(Desktop);
 Desktop.ParseDisplayName(0, nil, P, Len, Machine, Flags);
 Desktop.BindToObject(Machine, nil, IShellFolder, Pointer(Result));
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
 Memo3.Lines.Clear;
 Screen.Cursor:=crHourGlass;
 try
  if not EnumSharedResources(Edit1.Text, Memo3.Lines)
   then raise Exception.Create(‘ Невірне ім"я комп"ютера’ );
 finally
  Screen.Cursor:=crDefault;
 end;
end;



procedure TForm1.ClickMeClick(Sender: TObject);
var
 N: TNetworkNeighborhood;
 List: TStringList;
 i, j: Integer;
 ListViewItem: TListItem;
 WorkgroupNode, ComputerNode: TTreeNode;
begin
 Screen.Cursor:=crHourGlass;
 try
  // Ініціалізація об’ єктів та сканування мережі з використанням стандарту WiFi
6
  N:=TNetworkNeighborhood.Create;
  try
   // Отримання списку всіх комп’ ютерів в мережі з використанням стандарту WiFi
6
   Memo1.Lines.Clear;
   N.ListComputers(Memo1.Lines);

   // Отримання списку всіх робочих груп і комп’ ютерів в мережі з використанням
стандарту WiFi 6, відсортованих в алфавітному порядку
   List:=TStringList.Create;
   ListView1.Items.Clear;
   try
    N.ListNetwork(List);
    for i:=0 to List.Count - 1 do begin
     ListViewItem:=ListView1.Items.Add;
     ListViewItem.Caption:=List[i];
     ListViewItem.ImageIndex:=Integer(List.Objects[i]);
    end;
   finally
    List.Free;
   end;
```

```
   // Побудова дерева робочих груп і комп’ ютерів в мережі з використанням
стандарту WiFi 6
   TreeView1.Items.Clear;
   for i:=0 to N.Count - 1 do begin
    WorkgroupNode:=TreeView1.Items.Add(nil, N[i]);
    WorkgroupNode.ImageIndex:=1;
    WorkgroupNode.SelectedIndex:=1;
    for j:=0 to (N.Objects[i] as TStrings).Count - 1 do begin
     ComputerNode:=TreeView1.Items.AddChild(WorkgroupNode, (N.Objects[i] as
TStrings).Strings[j]);
     ComputerNode.ImageIndex:=0;
    end;
   end;

  // Отримання IP адрес комп’ ютерів
  GetIPAddresses(N, Memo2.Lines);

  finally
   N.Free;
  end;
  TreeView1.FullExpand;

 finally
  Screen.Cursor:=crDefault;
 end;
end;


procedure TForm1.FormCreate(Sender: TObject);
begin
 Edit1.Text:=GetComputerName;

  if LoadIpHlp then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end;

end;


procedure TForm1.Timer1Timer(Sender: TObject);
begin

   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;

end;


procedure TForm1.DOIpStuff;
begin

 Get_TCPTable( MemoX.Lines );
 Get_UDPTable( MemoTR.Lines );

end;


procedure TForm1.Button2Click(Sender: TObject);
begin

Form2.Show;

end;
```

```
procedure TForm1.Button3Click(Sender: TObject);
begin

Form1.Close;

end;


procedure TForm1.SpeedButton2Click(Sender: TObject);
begin

Form1.Close;

end;

procedure TForm1.SpeedButton4Click(Sender: TObject);
begin

Form2.Show;

end;


end.
```

Файл IPtraff.pas – відслідковування та контроль трафіку в мережі з використанням стандарту WiFi 6

```
unit IPtraff;

interface

uses
  Windows, Messages, SysUtils, Classes, Dialogs, IpHlpApi;

const
  NULL_IP       = '   0.  0.  0.  0' ;


type
  TWellKnownPort = record
    Prt: DWORD;
    Srv: string[20];
  end;


const

  WellKnownPorts: array[1..32] of TWellKnownPort
  = (
//    ( Prt: 0; Srv:   ' RESRVED'  ),     {Зарезервовано}
    ( Prt: 7; Srv:   ' ECHO   '  ),     {Пінгування   }
    ( Prt: 9; Srv:   ' DISCARD'  ),
    ( Prt: 13; Srv:  ' DAYTIME'  ),
    ( Prt: 17; Srv:  ' QOTD   '  ),     {Показчик на день}
    ( Prt: 19; Srv:  ' CHARGEN'  ),     {Генератор символів}
    ( Prt: 20; Srv:  ' FTPDATA'  ),     { Протокол File Transfer Protocol -
дані}
    ( Prt: 21; Srv:  ' FTPCTRL'  ),     { Протокол File Transfer Protocol -
управління}
    ( Prt: 22; Srv:  ' SSH    '  ),
    ( Prt: 23; Srv:  ' TELNET '  ),
    ( Prt: 25; Srv:  ' SMTP   '  ),     { Протокол Simple Mail Transfer
Protocol}
    ( Prt: 37; Srv:  ' TIME   '  ),     { Протокол Time Protocol }
    ( Prt: 43; Srv:  ' WHOIS  '  ),     { WHO IS сервіс  }
    ( Prt: 53; Srv:  ' DNS    '  ),     { Domain Name Service }
    ( Prt: 67; Srv:  ' BOOTPS '  ),     { BOOTP сервер }
    ( Prt: 68; Srv:  ' BOOTPC '  ),     { BOOTP клієнт }
    ( Prt: 69; Srv:  ' TFTP   '  ),     { Стандартний  FTP  }
    ( Prt: 70; Srv:  ' GOPHER '  ),     { Протокол Gopher       }
    ( Prt: 79; Srv:  ' FINGER '  ),     { Протокол Finger       }
    ( Prt: 80; Srv:  ' HTTP   '  ),     { Протокол HTTP         }
    ( Prt: 88; Srv:  ' KERBROS'  ),     { Протокол Kerberos     }
    ( Prt: 109; Srv: ' POP2   '  ),     { Протокол Post Office Protocol Version
2 }
    ( Prt: 110; Srv: ' POP3   '  ),     { Протокол Post Office Protocol Version
3 }
    ( Prt: 111; Srv: ' SUN_RPC'  ),     { SUN віддалений виклик функцій }
    ( Prt: 119; Srv: ' NNTP   '  ),     { Протокол Network News Transfer
Protocol }
    ( Prt: 123; Srv: ' NTP    '  ),     { Протокол Network Time protocol
}
    ( Prt: 135; Srv: ' DCOMRPC'  ),     { Локальний сервіce             }
    ( Prt: 137; Srv: ' NBNAME '  ),     { NETBIOS сервіс імен          }
    ( Prt: 138; Srv: ' NBDGRAM'  ),     { NETBIOS сервіс датаграм     }
    ( Prt: 139; Srv: ' NBSESS '  ),     { NETBIOS сервіс сесій        }
    ( Prt: 143; Srv: ' IMAP   '  ),     { Протокол Internet Message Access
Protocol }
    ( Prt: 161; Srv: ' SNMP   '  ),     { Протокол Simple Netw. Management
Protocol }
    ( Prt: 169; Srv: ' SEND   '  )
    );
```

```
  const
  ICMP_ERROR_BASE = 11000;
  IcmpErr : array[1..22] of string =
  (
   ' IP_BUFFER_TOO_SMALL' ,' IP_DEST_NET_UNREACHABLE' , '
IP_DEST_HOST_UNREACHABLE' ,
   ' IP_PROTOCOL_UNREACHABLE' , ' IP_DEST_PORT_UNREACHABLE' , ' IP_NO_RESOURCES'
,
   ' IP_BAD_OPTION' ,' IP_HARDWARE_ERROR' , ' IP_PACKET_TOO_BIG' , '
IP_REQUEST_TIMED_OUT' ,
   ' IP_BAD_REQUEST' ,' IP_BAD_ROUTE' , ' IP_TTL_EXPIRED_TRANSIT' ,
   ' IP_TTL_EXPIRED_REASSEM' ,' IP_PARAMETER_PROBLEM' , ' IP_SOURCE_QUENCH' ,
   ' IP_OPTION_TOO_BIG' , ' IP_BAD_DESTINATION' ,' IP_ADDRESS_DELETED' ,
   ' IP_SPEC_MTU_CHANGE' , ' IP_MTU_CHANGE' , ' IP_UNLOAD'
  );


  ARPEntryType  : array[1..4] of string = ( ' Інший' , ' Несправний' ,
    ' Динамічний' , ' Статичний'
    );
  TCPConnState  :
    array[1..12] of string =
 ( ' CLOSED' , ' READ' , ' SYN_SENT' ,
    ' SYN_RCVD' , ' ESTABLISHED' , ' FIN_WAIT1' ,
    ' FIN_WAIT2' , ' WAIT' , ' CLOSING' ,
    ' LAST_ACK' , ' WAIT' , ' DELETE_TCB'    );

  TCPToAlgo     : array[1..4] of string =
    ( ' Const.Timeout' , ' MIL-STD-1778' ,
    ' Van Jacobson' , ' Other'  );

  IPForwTypes   : array[1..4] of string =
    ( ' other' , ' invalid' , ' local' , ' remote'  );

  IPForwProtos  : array[1..18] of string =
    ( ' OTHER' , ' LOCAL' , ' NETMGMT' , ' ICMP' , ' EGP' ,
    ' GGP' , ' HELO' , ' RIP' , ' IS_IS' , ' ES_IS' ,
    ' CISCO' , ' BBN' , ' OSPF' , ' BGP' , ' BOOTP' ,
    ' AUTO_STAT' , ' STATIC' , ' NOT_DOD'  );

type

// для IpHlpNetworkParams
  TNetworkParams = record
    HostName: string ;
    DomainName: string ;
    CurrentDnsServer: string ;
    DnsServerTot: integer ;
    DnsServerNames: array [0..9] of string ;
    NodeType: UINT;
    ScopeID: string ;
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;

  TIfRows = array of TMibIfRow ; // динамічний масив колонок

// для IpHlpAdaptersInfo
  TAdaptorInfo = record
    AdapterName: string ;
    Description: string ;
    MacAddress: string ;
    Index: DWORD;
    aType: UINT;
    DHCPEnabled: UINT;
    CurrIPAddress: string ;
    CurrIPMask: string ;
```

```
    IPAddressTot: integer ;
    IPAddressList: array of string ;
    IPMaskList: array of string ;
    GatewayTot: integer ;
    GatewayList: array of string ;
    DHCPTot: integer ;
    DHCPServer: array of string ;
    HaveWINS: BOOL;
    PrimWINSTot: integer ;
    PrimWINSServer: array of string ;
    SecWINSTot: integer ;
    SecWINSServer: array of string ;
    LeaseObtained: LongInt ;
    LeaseExpires: LongInt;
  end ;

  TAdaptorRows = array of TAdaptorInfo ;



function IpHlpAdaptersInfo(var AdpTot: integer;var AdpRows: TAdaptorRows):
integer ;
procedure Get_AdaptersInfo( List: TStrings );
function IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
procedure Get_NetworkParams( List: TStrings );
procedure Get_ARPTable( List: TStrings );
procedure Get_TCPTable( List: TStrings );
procedure Get_TCPStatistics( List: TStrings );
function IpHlpTCPStatistics (var TCPStats: TMibTCPStats): integer ;
procedure Get_UDPTable( List: TStrings );
procedure Get_UDPStatistics( List: TStrings );
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;
procedure Get_IPAddrTable( List: TStrings );
procedure Get_IPForwardTable( List: TStrings );
procedure Get_IPStatistics( List: TStrings );
function IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint;
  var RTT: longint; var HopCount: longint ): integer;
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
function IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
procedure Get_IfTable( List: TStrings );
function IpHlpIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
procedure Get_RecentDestIPs( List: TStrings );

// утіліти перетворення
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
function IpAddr2Str( IPAddr: DWORD ): string;
function Str2IpAddr( IPStr: string ): DWORD;
function Port2Str( nwoPort: DWORD ): string;
function Port2Wrd( nwoPort: DWORD ): DWORD;
function Port2Svc( Port: DWORD ): string;
function ICMPErr2Str( ICMPErrCode: DWORD) : string;

implementation

var
  RecentIPs     : TStringList;


{ перетворення точена у рядок, потім рядок знищується }
function NextToken( var s: string; Separator: char ): string;
var
  Sep_Pos       : byte;
begin
  Result := ' ' ;
  if length( s ) > 0 then begin
    Sep_Pos := pos( Separator, s );
    if Sep_Pos > 0 then begin
```

```
      Result := copy( s, 1, Pred( Sep_Pos ) );
      Delete( s, 1, Sep_Pos );
    end
    else begin
      Result := s;
      s := ' ' ;
    end;
  end;
end;


{ пертворення MAC-адреси до ww-xx-yy-zz рядка }
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
var
  i             : integer;
begin
  if Size = 0 then
  begin
    Result := ' 00-00-00-00-00-00' ;
    EXIT;
  end
  else Result := ' ' ;
  //
  for i := 1 to Size do
    Result := Result + IntToHex( MacAddr[i], 2 ) + ' -' ;
  Delete( Result, Length( Result ), 1 );
end;


{ перетворення  IP-адреси  в мережний байт типу DWORD }
function IpAddr2Str( IPAddr: DWORD ): string;
var
  i             : integer;
begin
  Result := ' ' ;
  for i := 1 to 4 do
  begin
    Result := Result + Format( ' %3d.' , [IPAddr and $FF] );
    IPAddr := IPAddr shr 8;
  end;
  Delete( Result, Length( Result ), 1 );
end;


{ перетворення  крапкову десяткову IP-адресу  в мережний байт типу  DWORD}
function Str2IpAddr( IPStr: string ): DWORD;
var
  i             : integer;
  Num           : DWORD;
begin
  Result := 0;
  for i := 1 to 4 do
  try
    Num := ( StrToInt( NextToken( IPStr, ' .'  ) ) ) shl 24;
    Result := ( Result shr 8 ) or Num;
  except
    Result := 0;
  end;

end;


{ перетворення  номер порту в мережний байт типу  DWORD }
function Port2Wrd( nwoPort: DWORD ): DWORD;
begin
  Result := Swap( WORD( nwoPort ) );
end;
```

```
{ перетворення  номера порту в мережний байт типу  string }
function Port2Str( nwoPort: DWORD ): string;
begin
  Result := IntToStr( Port2Wrd( nwoPort ) );
end;


{ перетворення  відомого номера порту у сервіс ID }
function Port2Svc( Port: DWORD ): string;
var
  i             : integer;
begin
  Result := Format( ' %4d' , [Port] ); // у випадку якщо порт не знайдено
  for i := Low( WellKnownPorts ) to High( WellKnownPorts ) do
    if Port = WellKnownPorts[i].Prt then
    begin
      Result := WellKnownPorts[i].Srv;
      BREAK;
    end;
end;


{ фіксуємо параметри мережі з використанням стандарту WiFi 6 }

procedure Get_NetworkParams( List: TStrings );
var
    NetworkParams: TNetworkParams ;
    I, ErrorCode: integer ;
begin
    if not Assigned( List ) then EXIT;
    List.Clear;
    ErrorCode := IpHlpNetworkParams (NetworkParams) ;
    if ErrorCode <> 0 then
    begin
        List.Add (SysErrorMessage (ErrorCode));
        exit;
    end ;
    with NetworkParams do
    begin
        List.Add( ' HOSTNAME          : '  + HostName );
        List.Add( ' DOMAIN            : '  + DomainName );
        List.Add( ' NETBIOS NODE TYPE : '  + NETBIOSTypes[NodeType] );
        List.Add( ' DHCP SCOPE        : '  + ScopeID );
        List.Add( ' ROUTING ENABLED   : '  + IntToStr( EnableRouting ) );
        List.Add( ' PROXY   ENABLED   : '  + IntToStr( EnableProxy ) );
        List.Add( ' DNS     ENABLED   : '  + IntToStr( EnableDNS ) );
        if DnsServerTot <> 0 then
        begin
          for I := 0 to Pred (DnsServerTot) do
          List.Add( ' DNS SERVER ADDR   : '  + DnsServerNames [I] ) ;
        end ;
    end ;
end ;

//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
function IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
var
  FixedInfo     : PTFixedInfo;         //
  InfoSize      : Longint;
  PDnsServer    : PTIP_ADDR_STRING ;   //
begin
    InfoSize := 0 ;   //
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetNetworkParams( Nil, @InfoSize );  //
    if result <> ERROR_BUFFER_OVERFLOW then exit ; //
    GetMem (FixedInfo, InfoSize) ;                    //
    try
    result := GetNetworkParams( FixedInfo, @InfoSize );   //
```

```
    if result <> ERROR_SUCCESS then exit ;
    NetworkParams.DnsServerTot := 0 ;
    with FixedInfo^ do
    begin
        NetworkParams.HostName := trim (HostName) ;
        NetworkParams.DomainName := trim (DomainName) ;
        NetworkParams.ScopeId := trim (ScopeID) ;
        NetworkParams.NodeType := NodeType ;
        NetworkParams.EnableRouting := EnableRouting ;
        NetworkParams.EnableProxy := EnableProxy ;
        NetworkParams.EnableDNS := EnableDNS ;
        NetworkParams.DnsServerNames [0] := DNSServerList.IPAddress ;  //
        if NetworkParams.DnsServerNames [0] <> ' '  then
                                        NetworkParams.DnsServerTot := 1 ;
        PDnsServer := DnsServerList.Next;
        while PDnsServer <> Nil do
        begin
            NetworkParams.DnsServerNames [NetworkParams.DnsServerTot] :=
                                        PDnsServer^.IPAddress ;  //
            inc (NetworkParams.DnsServerTot) ;
            if NetworkParams.DnsServerTot >=
                            Length (NetworkParams.DnsServerNames) then exit ;
            PDnsServer := PDnsServer.Next ;
        end;
    end ;
    finally
       FreeMem (FixedInfo) ;                    //
    end ;
end;

//---------------------------------------------------------------------------

function ICMPErr2Str( ICMPErrCode: DWORD) : string;
begin
   Result := ' UnknownError : ' + IntToStr( ICMPErrCode );
   dec( ICMPErrCode, ICMP_ERROR_BASE );
   if ICMPErrCode in [Low(ICMpErr)..High(ICMPErr)] then
     Result := ICMPErr[ ICMPErrCode];
end;


//---------------------------------------------------------------------------

// включаємо байти вводу/виводу для кожного адаптеру

function IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
var
  I,
  TableSize  : integer;
  pBuf, pNext : PChar;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadIpHlp then exit ;
  SetLength (IfRows, 0) ;
  IfTot := 0 ; //
  TableSize := 0;
   //  перший виклик: беремо необхідний розмір пам' яті
  result := GetIfTable (Nil, @TableSize, false) ;  //
  if result <> ERROR_INSUFFICIENT_BUFFER then exit ;
  GetMem( pBuf, TableSize );
  try
      FillChar (pBuf^, TableSize, #0);  // очищуємо буфер, з W98 не потрібно

   // беремо показчик на таблицю
      result := GetIfTable (PTMibIfTable (pBuf), @TableSize, false) ;
      if result <> NO_ERROR then exit ;
      IfTot := PTMibIfTable (pBuf)^.dwNumEntries ;
      if IfTot = 0 then exit ;
      SetLength (IfRows, IfTot) ;
```

```
      pNext := pBuf + SizeOf(IfTot) ;
      for i := 0 to Pred (IfTot) do
      begin
         IfRows [i] := PTMibIfRow (pNext )^ ;
         inc (pNext, SizeOf (TMibIfRow)) ;
      end;
  finally
      FreeMem (pBuf) ;
  end ;
end;

procedure Get_IfTable( List: TStrings );
var
  IfRows        : TIfRows ;
  Error, I      : integer;
  NumEntries    : integer;
  sDescr, sIfName: string ;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (IfRows, 0) ;
  Error := IpHlpIfTable (NumEntries, IfRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if NumEntries = 0 then
      List.Add( ' Даних немає.'  )
  else
  begin
      for I := 0 to Pred (NumEntries) do
      begin
          with IfRows [I] do
          begin
             if wszName [1] = #0 then
                 sIfName := ' '
             else
                 sIfName := WideCharToString (@wszName) ;  // перетворюємо
Unicode у string
             sIfName := trim (sIfName) ;
             sDescr := bDescr ;
             sDescr := trim (sDescr);
             List.Add (Format (
               ' %0.8x - %3d - %16s - %8d - %12d - %2d - %2d - %10d - %10d  -
%-s - %-s' ,
              [dwIndex, dwType, MacAddr2Str( TMacAddress( bPhysAddr ),
              dwPhysAddrLen ), dwMTU, dwSpeed, dwAdminStatus,
              dwOPerStatus, Int64 (dwInOctets), Int64 (dwOutOctets),  //
counters are 32-bit
              sIfName, sDescr] )  //  , додаємо введення/вивід
              );
          end;
      end ;
  end ;
  SetLength (IfRows, 0) ;  // звільняємо пам' ять
end ;

function IpHlpIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadIpHlp then exit ;
  FillChar (IfRow, SizeOf (TMibIfRow), #0);  // очищуємо буфер, з W98 не
потрібно
  IfRow.dwIndex := Index ;
  result := GetIfEntry (@IfRow) ;
end ;

//---------------------------------------------------------------------------
{ інформація про мережні адаптери }
```

```
function IpHlpAdaptersInfo(var AdpTot: integer; var AdpRows: TAdaptorRows):
integer ;
var
  BufLen        : DWORD;
  AdapterInfo   : PTIP_ADAPTER_INFO;
  PIpAddr       : PTIP_ADDR_STRING;
  PBuf          : PCHAR ;
  I             : integer ;
begin
  SetLength (AdpRows, 4) ;
  AdpTot := 0 ;
  BufLen := 0 ;
  result := GetAdaptersInfo( Nil, @BufLen );
  if (result <> ERROR_INSUFFICIENT_BUFFER) and (result = NO_ERROR) then exit ;
  GetMem( pBuf, BufLen );
  try
      FillChar (pBuf^, BufLen, #0);  // очищуємо буфер
      result := GetAdaptersInfo( PTIP_ADAPTER_INFO (PBuf), @BufLen );
      if result = NO_ERROR then
      begin
         AdapterInfo := PTIP_ADAPTER_INFO (PBuf) ;
         while ( AdapterInfo <> nil ) do
         begin
            AdpRows [AdpTot].IPAddressTot := 0 ;
            SetLength (AdpRows [AdpTot].IPAddressList, 2) ;
            SetLength (AdpRows [AdpTot].IPMaskList, 2) ;
            AdpRows [AdpTot].GatewayTot := 0 ;
            SetLength (AdpRows [AdpTot].GatewayList, 2) ;
            AdpRows [AdpTot].DHCPTot := 0 ;
            SetLength (AdpRows [AdpTot].DHCPServer, 2) ;
            AdpRows [AdpTot].PrimWINSTot := 0 ;
            SetLength (AdpRows [AdpTot].PrimWINSServer, 2) ;
            AdpRows [AdpTot].SecWINSTot := 0 ;
            SetLength (AdpRows [AdpTot].SecWINSServer, 2) ;
            AdpRows [AdpTot].CurrIPAddress := NULL_IP;
            AdpRows [AdpTot].CurrIPMask := NULL_IP;
            AdpRows [AdpTot].AdapterName := Trim( string(
AdapterInfo^.AdapterName ) );
            AdpRows [AdpTot].Description := Trim( string(
AdapterInfo^.Description ) );
            AdpRows [AdpTot].MacAddress := MacAddr2Str( TMacAddress(
                                  AdapterInfo^.Address ),
AdapterInfo^.AddressLength ) ;
            AdpRows [AdpTot].Index := AdapterInfo^.Index ;
            AdpRows [AdpTot].aType := AdapterInfo^.aType ;
            AdpRows [AdpTot].DHCPEnabled := AdapterInfo^.DHCPEnabled ;
            if AdapterInfo^.CurrentIPAddress <> Nil then
            begin
                AdpRows [AdpTot].CurrIPAddress :=
AdapterInfo^.CurrentIPAddress.IpAddress ;
                AdpRows [AdpTot].CurrIPMask :=
AdapterInfo^.CurrentIPAddress.IpMask ;
            end ;

        // беремо список IP адрес та масок для IPAddressList
            I := 0 ;
            PIpAddr := @AdapterInfo^.IPAddressList ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].IPAddressList [I] := PIpAddr.IpAddress ;
                AdpRows [AdpTot].IPMaskList [I] := PIpAddr.IpMask ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
                if Length (AdpRows [AdpTot].IPAddressList) <= I then
                begin
                     SetLength (AdpRows [AdpTot].IPAddressList, I * 2) ;
                     SetLength (AdpRows [AdpTot].IPMaskList, I * 2) ;
                end ;
            end ;
```

```
            AdpRows [AdpTot].IPAddressTot := I ;

        // беремо список IP адрес для GatewayList
            I := 0 ;
            PIpAddr := @AdapterInfo^.GatewayList ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].GatewayList [I] := PIpAddr.IpAddress ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
                if Length (AdpRows [AdpTot].GatewayList) <= I then
                            SetLength (AdpRows [AdpTot].GatewayList, I * 2) ;
            end ;
            AdpRows [AdpTot].GatewayTot := I ;

        // беремо список IP адрес для GatewayList
            I := 0 ;
            PIpAddr := @AdapterInfo^.DHCPServer ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].DHCPServer [I] := PIpAddr.IpAddress ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
                if Length (AdpRows [AdpTot].DHCPServer) <= I then
                            SetLength (AdpRows [AdpTot].DHCPServer, I * 2) ;
            end ;
            AdpRows [AdpTot].DHCPTot := I ;

        // беремо список IP адрес для PrimaryWINSServer
            I := 0 ;
            PIpAddr := @AdapterInfo^.PrimaryWINSServer ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].PrimWINSServer [I] := PIpAddr.IpAddress ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
                if Length (AdpRows [AdpTot].PrimWINSServer) <= I then
                            SetLength (AdpRows [AdpTot].PrimWINSServer, I * 2)
;
            end ;
            AdpRows [AdpTot].PrimWINSTot := I ;

       // беремо список IP адрес для SecondaryWINSServer
            I := 0 ;
            PIpAddr := @AdapterInfo^.SecondaryWINSServer ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].SecWINSServer [I] := PIpAddr.IpAddress ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
                if Length (AdpRows [AdpTot].SecWINSServer) <= I then
                            SetLength (AdpRows [AdpTot].SecWINSServer, I * 2) ;
            end ;
            AdpRows [AdpTot].SecWINSTot := I ;

            AdpRows [AdpTot].LeaseObtained := AdapterInfo^.LeaseObtained ;
            AdpRows [AdpTot].LeaseExpires := AdapterInfo^.LeaseExpires ;

            inc (AdpTot) ;
            if Length (AdpRows) <= AdpTot then
                        SetLength (AdpRows, AdpTot * 2) ;  // more memory
            AdapterInfo := AdapterInfo^.Next;
         end ;
         SetLength (AdpRows, AdpTot) ;
      end ;
  finally
      FreeMem( pBuf );
  end ;
end ;
```

```
procedure Get_AdaptersInfo( List: TStrings );
var
  AdpTot: integer;
  AdpRows: TAdaptorRows ;
  Error: DWORD ;
  I: integer ;
  //J: integer ;  jpt
  //S: string ;        id.
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (AdpRows, 0) ;
  AdpTot := 0 ;
  Error := IpHlpAdaptersInfo(AdpTot, AdpRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if AdpTot = 0 then
      List.Add( ' Даних немає.'  )
  else
  begin
      for I := 0 to Pred (AdpTot) do
      begin
        with AdpRows [I] do
        begin
            //List.Add(AdapterName + ' -'  + Description ); // jpt : не корисне
            List.Add( Format(' %8.8x - %6s - %16s - %2d - %16s - %16s - %16s' ,
                [Index, AdaptTypes[aType], MacAddress, DHCPEnabled,
                GatewayList [0], DHCPServer [0], PrimWINSServer [0]] ) );
            {if IPAddressTot <> 0 then    // jpt : not useful
            begin
                S := ' '  ;
                for J := 0 to Pred (IPAddressTot) do
                        S := S + IPAddressList [J] + ' /'  + IPMaskList [J] + '
- '  ;
                List.Add(IntToStr (IPAddressTot) + '  IP Addresse(s): '  + S);
            end ;
            List.Add( '   '  ); }
        end ;
      end ;
  end ;
  SetLength (AdpRows, 0) ;
end ;

//---------------------------------------------------------------------------
{ зчитуємо кількість раундів, та час звертання до IP }
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint; var RTT: Longint;
  var HopCount: Longint ): integer;
begin
  if not GetRTTAndHopCount( IPAddr, @HopCount, MaxHops, @RTT ) then
  begin
    Result := GetLastError;
    RTT := -1; //
    HopCount := -1;
  end
  else
    Result := NO_ERROR;
end;

//---------------------------------------------------------------------------
{ ARP-таблиця – список відношень між віддаленими IP та віддаленими MAC-адресами.

}
procedure Get_ARPTable( List: TStrings );
var
  IPNetRow      : TMibIPNetRow;
  TableSize     : DWORD;
  NumEntries    : DWORD;
  ErrorCode     : DWORD;
```

```
  i             : integer;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  //  перший виклик: беремо довжину таблиці
  TableSize := 0;
  ErrorCode := GetIPNetTable( Nil, @TableSize, false );   //
  //
  if ErrorCode = ERROR_NO_DATA then
  begin
    List.Add( '  ARP-кеш пустий.'  );
    EXIT;
  end;
  // беремо таблицю
  GetMem( pBuf, TableSize );
  NumEntries := 0 ;
  try
  ErrorCode := GetIpNetTable( PTMIBIPNetTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBIPNetTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then //.
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        IPNetRow := PTMIBIPNetRow( PBuf )^;
        with IPNetRow do
          List.Add( Format( ' %8x - %12s - %16s - %10s' ,
                            [dwIndex, MacAddr2Str( bPhysAddr, dwPhysAddrLen ),
                            IPAddr2Str( dwAddr ), ARPEntryType[dwType]
                            ]));
        inc( pBuf, SizeOf( IPNetRow ) );
      end;
    end
    else
      List.Add( '  ARP-кеш пустий.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // we _must_ restore pointer!
  finally
      dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPNetRow ) );
      FreeMem( pBuf );
  end ;
end;


//-----------------------------------------------------------------------------
procedure Get_TCPTable( List: TStrings );
var
  TCPRow        : TMIBTCPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  DestIP        : string;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  RecentIPs.Clear;
  //  перший виклик : беремо розмір таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetTCPTable(Nil, @TableSize, false );  //
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
```

```
    EXIT;

  // беремо розмір пам' яти, який потрібно та здійснюємо виклик знову
  GetMem( pBuf, TableSize );
  // беремо таблицю
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin

    NumEntries := PTMIBTCPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        TCPRow := PTMIBTCPRow( pBuf )^; // беремо наступний запис
        with TCPRow do
        begin
          if dwRemoteAddr = 0 then
            dwRemotePort := 0;
          DestIP := IPAddr2Str( dwRemoteAddr );
          List.Add(
            Format( ' %15s : %-7s -> %15s : %-7s -> %-16s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) ),
              DestIP,
              Port2Svc( Port2Wrd( dwRemotePort ) ),
              TCPConnState[dwState]
              ] ) );
         //
            if (not ( dwRemoteAddr = 0 ))
            and ( RecentIps.IndexOf(DestIP) = -1 ) then
               RecentIPs.Add( DestIP );
        end;
        inc( pBuf, SizeOf( TMIBTCPRow ) );
      end;
    end;
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibTCPRow ) );
  FreeMem( pBuf );
end;

//---------------------------------------------------------------------------
procedure Get_TCPStatistics( List: TStrings );
var
  TCPStats     : TMibTCPStats;
  ErrorCode    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  if NOT LoadIpHlp then exit ;
  ErrorCode := GetTCPStatistics( @TCPStats );
  if ErrorCode = NO_ERROR then
    with TCPStats do
    begin
      List.Add( ' Алгоритм повторної передачі: '  + TCPToAlgo[dwRTOAlgorithm] );
      List.Add( ' Мінімальний час виведення         : '  + IntToStr( dwRTOMin )
+ '  ms'  );
      List.Add( ' Максимальний час виведення         : '  + IntToStr( dwRTOMax )
+ '  ms'  );
      List.Add( ' Максимальне число підключень : '  + IntToStr( dwRTOAlgorithm )
);
      List.Add( ' Активні підключення             : '  + IntToStr( dwActiveOpens
) );
      List.Add( ' Пасивні підключення            : '  + IntToStr( dwPassiveOpens
) );
```

```
      List.Add( ' Невдала спроба відкриття     : '  + IntToStr( dwAttemptFails )
);
      List.Add( ' Відновлений зв'язок   : '  + IntToStr( dwEstabResets ) );
      List.Add( ' Поточний зв'язок .: '  + IntToStr( dwCurrEstab ) );
      List.Add( ' Отримано сегментів        : '  + IntToStr( dwInSegs ) );
      List.Add( ' Відправлено сегментів            : '  + IntToStr( dwOutSegs )
);
      List.Add( ' Ретрансльовано сегментів   : '  + IntToStr( dwReTransSegs ) );
      List.Add( ' Помилки входження          : '  + IntToStr( dwInErrs ) );
      List.Add( ' Скидання виведення          : '  + IntToStr( dwOutRsts ) );
      List.Add( ' Сукупні зв'язки   : '  + IntToStr( dwNumConns ) );
    end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
end;

function IpHlpTCPStatistics (var TCPStats: TMibTCPStats): integer ;
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetTCPStatistics( @TCPStats );
end;

//-------------------------------------------------------------------------
procedure Get_UDPTable( List: TStrings );
var
  UDPRow        : TMIBUDPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;

  //  перший виклик : беремо розмір таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetUDPTable(Nil, @TableSize, false );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо потрібний розмір пам'яті, викликаємо знову
  GetMem( pBuf, TableSize );

  // беремо таблицю
  ErrorCode := GetUDPTable( PTMIBUDPTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBUDPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        UDPRow := PTMIBUDPRow( pBuf )^; // беремо наступний запис
        with UDPRow do
          List.Add( Format( ' %15s : %-6s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) )
              ] ) );
        inc( pBuf, SizeOf( TMIBUDPRow ) );
      end;
    end
    else
      List.Add( ' Даних немає.'  );
  end
  else
```

```
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibUDPRow ) );
  FreeMem( pBuf );
end;

//-------------------------------------------------------------------------
procedure Get_IPAddrTable( List: TStrings );
var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0; ;
  NumEntries := 0 ;
  //  перший виклик: беремо довжину таблиці
  ErrorCode := GetIpAddrTable(Nil, @TableSize, true );  //
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // беремо таблицю
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPAddrTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPAddrRow := PTMIBIPAddrRow( pBuf )^;
        with IPAddrRow do
          List.Add( Format( ' %8.8x | %15s | %15s | %15s | %8.8d' ,
            [dwIndex,
            IPAddr2Str( dwAddr ),
              IPAddr2Str( dwMask ),
              IPAddr2Str( dwBCastAddr ),
              dwReasmSize
              ] ) );
        inc( pBuf, SizeOf( TMIBIPAddrRow ) );
      end;
    end
    else
      List.Add( ' Даних немає.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // we must restore pointer!
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;

//-------------------------------------------------------------------------
{ беремо дані з таблиці маршрутизатора мережі з використанням стандарту WiFi 6}
procedure Get_IPForwardTable( List: TStrings );
var
  IPForwRow     : TMibIPForwardRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
```

```
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0;

  //  перший виклик: беремо довжину таблиці
  NumEntries := 0 ;
  ErrorCode := GetIpForwardTable(Nil, @TableSize, true);
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо таблицю
  GetMem( pBuf, TableSize );
  ErrorCode := GetIpForwardTable( PTMibIPForwardTable( pBuf ), @TableSize,
true);
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPForwardTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPForwRow := PTMibIPForwardRow( pBuf )^;
        with IPForwRow do
        begin
          if (dwForwardType < 1)
          or (dwForwardType > 4) then
                   dwForwardType := 1 ;   //  , враховуємо погані значення
          List.Add( Format(
            ' %15s | %15s | %15s | %8.8x | %7s |   %5.5d |  %7s |   %2.2d' ,
            [IPAddr2Str( dwForwardDest ),
            IPAddr2Str( dwForwardMask ),
              IPAddr2Str( dwForwardNextHop ),
              dwForwardIFIndex,
              IPForwTypes[dwForwardType],
              dwForwardNextHopAS,
              IPForwProtos[dwForwardProto],
              dwForwardMetric1
              ] ) );
        end ;
        inc( pBuf, SizeOf( TMibIPForwardRow ) );
      end;
    end
    else
      List.Add( ' Даних немає.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibIPForwardRow ) );
  FreeMem( pBuf );
end;

//---------------------------------------------------------------------------
procedure Get_IPStatistics( List: TStrings );
var
  IPStats       : TMibIPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  if NOT LoadIpHlp then exit ;
  ErrorCode := GetIPStatistics( @IPStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with IPStats do
    begin
      if dwForwarding = 1 then
        List.add( ' Розблоковане пересилання      : '  + ' Так'  )
```

```
      else
        List.add( ' Розблоковане пересилання      : '  + ' Hi'  );
      List.add( ' Вбудований TTL             : '  + inttostr( dwDefaultTTL ) );
      List.add( ' Отримані датаграми       : '  + inttostr( dwInReceives ) );
      List.add( ' Помилка заголовку     (в)  : '  + inttostr( dwInHdrErrors ) );
      List.add( ' Адреса помилкова    (в)  : '  + inttostr( dwInAddrErrors ) );
      List.add( ' Датаграма переслана     : '  + inttostr( dwForwDatagrams ) );
//
      List.add( ' Невідомий протокол (в)  : '  + inttostr( dwInUnknownProtos )
);
      List.add( ' Датаграма відвергнута     : '  + inttostr( dwInDiscards ) );
      List.add( ' Датаграма встановлена     : '  + inttostr( dwInDelivers ) );
      List.add( ' Запит виведення           : '  + inttostr( dwOutRequests ) );
      List.add( ' Маршрутизація в маршрутизаторі мережі з використанням
стандарту WiFi 6 відвергнута      : '  + inttostr( dwRoutingDiscards ) );
      List.add( ' Немає маршрутів        (з)  : '  + inttostr( dwOutNoRoutes )
);
      List.add( ' Час виведення перебору      : '  + inttostr( dwReasmTimeOut )
);
      List.add( ' Запити перебору     : '  + inttostr( dwReasmReqds ) );
      List.add( ' Перебори здійснено : '  + inttostr( dwReasmOKs ) );
      List.add( ' Помилка перебору     : '  + inttostr( dwReasmFails ) );
      List.add( ' Фрагментація мережі з використанням стандарту WiFi 6 успішна:
'  + inttostr( dwFragOKs ) );
      List.add( ' Помилка фрагментації   : '  + inttostr( dwFragFails ) );
      List.add( ' Датаграми фрагментовано    : '  + inttostr( dwFRagCreates ) );
      List.add( ' Кількість інтерфейсів    : '  + inttostr( dwNumIf ) );
      List.add( ' Кількість IP-адрес  : '  + inttostr( dwNumAddr ) );
      List.add( ' маршрут в таблиці маршрутизації мережі з використанням
стандарту WiFi 6 : '  + inttostr( dwNumRoutes ) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

function IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;      //
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetIPStatistics( @IPStats );
end ;

//------------------------------------------------------------------------------
procedure Get_UdpStatistics( List: TStrings );
var
  UdpStats     : TMibUDPStats;
  ErrorCode    : integer;
begin
  if not Assigned( List ) then EXIT;
  ErrorCode := GetUDPStatistics( @UdpStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with UDPStats do
    begin
      List.add( ' Datagrams (в)    : '  + inttostr( dwInDatagrams ) );
      List.add( ' Datagrams (з)   : '  + inttostr( dwOutDatagrams ) );
      List.add( ' No Ports          : '  + inttostr( dwNoPorts ) );
      List.add( ' Errors    (в)    : '  + inttostr( dwInErrors ) );
      List.add( ' UDP Listen Ports  : '  + inttostr( dwNumAddrs ) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;
```

```
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;     //
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetUDPStatistics (@UdpStats) ;
end ;



//-------------------------------------------------------------------------------

procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );

var
  ErrorCode     : DWORD;
  ICMPStats     : PTMibICMPInfo;

begin
  if ( ICMPIn = nil ) or ( ICMPOut = nil ) then EXIT;
  ICMPIn.Clear;
  ICMPOut.Clear;
  New( ICMPStats );
  ErrorCode := GetICMPStatistics( ICMPStats );
  if ErrorCode = NO_ERROR then
  begin
    with ICMPStats.InStats do
    begin
      ICMPIn.Add( ' Отримано повідослень    : ' + IntToStr( dwMsgs ) );
      ICMPIn.Add( ' Помилки ICMP            : ' + IntToStr( dwErrors ) );
      ICMPIn.Add( ' Розташування    : ' + IntToStr( dwDestUnreachs ) );
      ICMPIn.Add( ' Час перевищено        : ' + IntToStr( dwTimeEcxcds ) );
      ICMPIn.Add( ' Проблеми параметрів      : ' + IntToStr( dwParmProbs ) );
      ICMPIn.Add( ' Джерело відключено       : ' + IntToStr( dwSrcQuenchs ) );
      ICMPIn.Add( ' Перенаправлення          : ' + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' Ехо запит          : ' + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Ехо повтор       : ' + IntToStr( dwEchoReps ) );
      ICMPIn.Add( ' Запит мітки часу   : ' + IntToStr( dwTimeStamps ) );
      ICMPIn.Add( ' Повтор мітки часу    : ' + IntToStr( dwTimeStampReps ) );
      ICMPIn.Add( ' Запит маски адреси : ' + IntToStr( dwAddrMasks ) );
      ICMPIn.Add( ' Повтор маски адреси   : ' + IntToStr( dwAddrReps ) );
    end;
     //


    with ICMPStats.OutStats do


 begin
      ICMPOut.Add( ' Повідомлення відправлено        : ' + IntToStr( dwMsgs )
);
      ICMPOut.Add( 'Помилки ICMP            : ' + IntToStr( dwErrors ) );
      ICMPOut.Add( 'Розташування    : ' + IntToStr( dwDestUnreachs ) );
      ICMPOut.Add( 'Час перевищено        : ' + IntToStr( dwTimeEcxcds ) );
      ICMPOut.Add( 'Проблеми параметрів      : ' + IntToStr( dwParmProbs ) );
      ICMPOut.Add( 'Джерело відключено       : ' + IntToStr( dwSrcQuenchs ) );
      ICMPOut.Add( 'Перенаправлення          : ' + IntToStr( dwRedirects ) );
      ICMPOut.Add( 'Ехо запит        : ' + IntToStr( dwEchos ) );
      ICMPOut.Add( 'Ехо повтор       : ' + IntToStr( dwEchoReps ) );
      ICMPOut.Add( 'Запит мітки часу   : ' + IntToStr( dwTimeStamps ) );
      ICMPOut.Add( 'Повтор мітки часу    : ' + IntToStr( dwTimeStampReps ) );
      ICMPOut.Add( 'Запит маски адреси: ' + IntToStr( dwAddrMasks ) );
      ICMPOut.Add( 'Повтор маски адреси   : ' + IntToStr( dwAddrReps ) );
    end;
  end
  else
    IcmpIn.Add( SysErrorMessage( ErrorCode ) );
  Dispose( ICMPStats );
```

```
end;


//---------------------------------------------------------------------------
procedure Get_RecentDestIPs( List: TStrings );
begin
  if Assigned( List ) then
    List.Assign( RecentIPs )
end;

initialization

  RecentIPs := TStringList.Create;

finalization

  RecentIPs.Free;

end.
```

Файл NetConst.pas – ініціалізація констант

```
unit NetConst;



interface

resourcestring

  SShellLinkReadError = ' Помилка  читання' ;
  SShellLinkWriteError = ' Помилка запису' ;
  SShellLinkLoadError = ' Не можу завантажити %s' ;
  SShellLinkSaveError = ' Не можу зберегти %s' ;
  SShellLinkCreateError = ' Не можу ініціалізувати інтерфейс' ;
  SDynArrayIndexError = ' У масиві %s немає елемента з таким індексом (%d)' ;
  SDynArrayCountError = ' Перевищена довжина масиву (%d)' ;
  SSharedMemoryError = ' Не можу створити файл' ;
  SCannotInitTimer = ' Не можу ініціалізувати таймер' ;
  SPrinterIndexError = ' Принтер не доступний (%d)' ;
  SIndicesOutOfRange = ' Недопустимий індекс матриці [%d: %d]' ;
  SRowIndexOutOfRange = ' Недопустиме занчення рядка матриці (%d)' ;
  SColIndexOutOfRange = ' Недопустиме занчення стовбчика матриці (%d)' ;
  SNoAdminRights = ' У Вас немає прав адміністратора' ;

  SFileError = ' Помилка %s файл %s%s' ;
  SFileReading = ' читання' ;
  SFileWriting = ' запис' ;

 SFileError002 = '  - файл не знайдено' ;
  SFileError003 = '  - шлях не знайдено' ;
  SFileError004 = '  - не можу відкрити файл' ;
  SFileError005 = '  - немає доступу' ;
  SFileError014 = '  - не достатньо пам"яті' ;
  SFileError015 = '  - не можу знайти драйвер' ;
  SFileError017 = '  - не можу перемістити файл' ;
  SFileError019 = '  - носій захищений від запису' ;
  SFileError020 = '  - не можу знайти пристрій' ;
  SFileError021 = '  - пристрій не відкривається для читання' ;
  SFileError022 = '  - пристрій не може розпізнати команду' ;
  SFileError025 = '  - вказана область не знайдена' ;
  SFileError026 = '  - пристрій недоступний' ;
  SFileError027 = '  - сектор не знайдено' ;
  SFileError029 = '  - помилка запису на пристрій' ;
  SFileError030 = '  - помилка читання з пристрою' ;
  SFileError032 = '  - файл використовується іншою програмою' ;
  SFileError036 = '  - занадто багато відкритих файлів' ;
  SFileError038 = '  - досягнутий кінець файлу' ;
  SFileError039 = '  - диск переповнений' ;
  SFileError050 = '  - запит не підтримується' ;
  SFileError051 = '  - віддалений комп' ютер недоступний' ;
  SFileError052 = '  - у мережі з використанням стандарту WiFi 6 знайдені
ідентичні імена' ;
  SFileError053 = '  - мережний шлях не знайдено' ;
  SFileError054 = '  – мережа зайнята' ;
  SFileError055 = '  - ресурс мережі з використанням стандарту WiFi 6 або
пристрій недоступний' ;
  SFileError057 = '  - апаратна помилка в мережному адаптері' ;
  SFileError058 = '  - сервер не в змозі виконувати дану дію' ;
  SFileError059 = '  - помилка у мережі з використанням стандарту WiFi 6' ;
  SFileError064 = '  - недоступне мережне ім"я' ;
  SFileError065 = '  - немає доступу до мережі з використанням стандарту WiFi 6'
;
  SFileError066 = '  - невірно вказатий тип мережного ресурсу' ;
  SFileError067 = '  - не знайдене вказане мережне ім"я' ;
  SFileError070 = '  - відключейний сервер мережі з використанням стандарту WiFi
6' ;
```

```
  SFileError082 = '  - не можу створити файл чи каталог' ;
  SFileError112 = '  - не достатньо вільного місця на диску' ;
  SFileError123 = '  - в імені файлу вказано недопустимий символ' ;
  SFileError161 = '  - неправильно вказано шдях' ;
  SFileError183 = '  - файл не існує' ;

  SCannotSetSize = ' Не можу змінити розмір файлу' ;

  SUnableToCompress = ' Не можу заархівувати дані' ;
  SUnableToDecompress = ' Не можу розархівувати дані' ;

  SCannotFindNetwork = ' Не можу знайти  мережу' ;

implementation

end.
```

Файл Networks.pas – робота з мережою з використанням стандарту WiFi 6

```
unit Networks;

interface

uses Windows, ShellAPI, ShlObj, ActiveX, ComObj, Dim, Classes, SysUtils,
WinSock;

type
  ComputerFound = class (Exception);
  ECannotFindNetwork = class (Exception);

  TStringObject = class (TObject)
  private
    FValue: TString;
    FTag: Integer;
    FData: Pointer;
    FRefObj: TObject;
    procedure SetValue(const Value: TString);
    procedure SetData(const Value: Pointer);
    procedure SetRefObj(const Value: TObject);
    procedure SetTag(const Value: Integer);
  public
    property Value: TString read FValue write SetValue;
    property RefObj: TObject read FRefObj write SetRefObj;
    property Tag: Integer read FTag write SetTag;
    property Data: Pointer read FData write SetData;
  end;

  TStringObjectArray = class (TDynamicArray)
  private
    function GetObject(Index: Integer): TStringObject;
    function GetData(Index: Integer): Pointer;
    function GetRefObj(Index: Integer): TObject;
    function GetTag(Index: Integer): Integer;
    function GetValue(Index: Integer): TString;
    procedure SetData(Index: Integer; const Value: Pointer);
    procedure SetRefObj(Index: Integer; const Value: TObject);
    procedure SetTag(Index: Integer; const Value: Integer);
    procedure SetValue(Index: Integer; const Value: TString);

    function FreeObject(Index: Integer; var Obj: TStringObject): Integer;

    procedure FreeItem(Index: Integer);
    procedure CreateItem(Index: Integer);

  protected
    procedure SetCount(const NewCount: Cardinal); override;
  public
    function Add: Integer; override;
    procedure Insert(Index: Integer); override;
    procedure Delete(Index: Integer); override;
    function AddItem(const Item): Integer; override;
    procedure InsertItem(Index: Integer; const Item); override;
    procedure DeleteItem(Index: Integer; out Item); override;
    property Value[Index: Integer]: TString read GetValue write SetValue;
default;
    property RefObj[Index: Integer]: TObject read GetRefObj write SetRefObj;
    property Tag[Index: Integer]: Integer read GetTag write SetTag;
    property Data[Index: Integer]: Pointer read GetData write SetData;
    constructor Create;
    destructor Destroy; override;
  end;

  TStringObjectList = class (TStrings)
  private
```

```
    FArray: TStringObjectArray;
    function GetData(Index: Integer): Pointer;
    function GetTag(Index: Integer): Integer;
    procedure SetData(Index: Integer; const Value: Pointer);
    procedure SetTag(Index: Integer; const Value: Integer);
  protected
    function Get(Index: Integer): string; override;
    function GetCount: Integer; override;
    function GetObject(Index: Integer): TObject; override;
    procedure Put(Index: Integer; const S: string); override;
    procedure PutObject(Index: Integer; AObject: TObject); override;

  public
    property Data[Index: Integer]: Pointer read GetData write SetData;
    property Tag[Index: Integer]: Integer read GetTag write SetTag;

    function Add(const S: string): Integer; override;
    procedure Clear; override;
    procedure Delete(Index: Integer); override;
    procedure Exchange(Index1, Index2: Integer); override;
    procedure Insert(Index: Integer; const S: string); override;


    constructor Create;
    destructor Destroy; override;
  end;

  {TNetworkWorkgroup - клас, який створює список всіх комп’ ютерів в робочій групі. Цей клас є нащадком класу TStrings і повністю сумісний з іншим нащадками цього класу.  Об’ єкти цього класу записуються у властивості об’ єктів класу TNetworkNeiborhood. Клас TNetworkNeiborhood містить об’ єкти і властивості робочих груп. }

  TNetworkWorkgroup = class (TStringObjectList);


  {TNetworkNeiborhood - клас, який створює список всіх робочих груп в мережі з використанням стандарту WiFi 6}
  TNetworkNeighborhood = class (TStringObjectList)
  private
    function CreatePIDL(Size: Integer): PItemIDList;
    procedure DisposePIDL(ID: PItemIDList);
    function NextPIDL(IDList: PItemIDList): PItemIDList;
    function GetPIDLSize(IDList: PItemIDList): Integer;
    function CopyPIDL(IDList: PItemIDList): PItemIDList;
    procedure StripLastID(IDList: PItemIDList);
    function GetPrevPIDL(PIDL: PItemIDList): PItemIDList;
    class function GetDisplayName(ShellFolder: IShellFolder; PIDL: PItemIDList): TString;
    function OriginFolder:  IShellFolder;
    function OriginFolderNT: IShellFolder;
    class function EnumObjects(ShellFolder: IShellFolder): IEnumIDList;
    class procedure ParseFolder(Folder: IShellFolder; Items: TStringObjectList; StorePIDLs: Boolean = False);
    class procedure ParseFolderEx(Folder: IShellFolder; Items: TStrings);

    function FreeRefObj(Index: Integer; var Obj: TStringObject): Integer;
    function GetWorkgroup(Name: TString): TNetworkWorkgroup;

  public
    { Процедура Оновлення шукає всі доступні робочі групи в мережі з використанням стандарту WiFi 6 }

    procedure Refresh;

    { містить списки всіх комп’ ютерів в  мережі з використанням стандарту WiFi 6}

    property Workgroup[Name: TString]: TNetworkWorkgroup read GetWorkgroup;
```

```
   { Функція FindComputer шукає комп’ ютер зі вказаним ім’ ям і повертає ім’ я
робочої групи де його знайдено, або порожню строку
     якщо комп’ ютера з таким іменем у мережі з використанням стандарту WiFi 6
немає }

    function FindComputer(Name: TString): TString;

    { Процедура ListComputers копіює список всіх комп’ ютерів в мережі з
використанням стандарту WiFi 6
      у об’ єкті TStrings}
    procedure ListComputers(Strings: TStrings);

    { Процедура ListNetwork копіює відсортований в алфавітному порядку список
всіх робочих груп і комп’ ютерів в мережі з використанням стандарту WiFi 6.
     Робочі групи мають ‘ TObject(1)’  у відповідному елементі властивості об’
єктів, а комп’ ютери - ‘ nil’ }

    procedure ListNetwork(Strings: TStrings);

    function Add(const S: string): Integer; override;
    procedure Clear; override;
    procedure Delete(Index: Integer); override;
    procedure Insert(Index: Integer; const S: string); override;
    constructor Create;
  end;

{ Функція GetIPAddress отримує IP адресу комп’ ютера або сервера.
  Параметр NetworkName конкретизує ім’ я комп’ ютера або  сервера.
  Ця функція повертає адреси IP у форматі XXX.XXX.XXX.XXX у разі успішного
виконання, ‘ Error’  - коли неможливо ініціалізувати, ‘ Unknown’  – коли
параметр NetworkName посилається на неіснуючий комп’ ютер або на комп’ ютер  без
встановленого протоколу TCP/IP }

  function GetIPAddress(NetworkName: TString): TString;


{ GetIPAddresses отримує IP адреси всіх доступних комп’ ютерів мережі з
використанням стандарту WiFi 6 }

procedure GetIPAddresses(Network: TNetworkNeighborhood; List: TStrings);



{ Функція EnumSharedResources перераховує загальні ресурси мережі з
використанням стандарту WiFi 6. Параметр ComputerName конкретизує
  ім’ я комп’ ютера. }

function EnumSharedResources(ComputerName: TString; List: TStrings): Boolean;


implementation

uses NetConst;

{ TStringObject }

procedure TStringObject.SetData(const Value: Pointer);
begin
  FData := Value;
end;

procedure TStringObject.SetRefObj(const Value: TObject);
begin
  FRefObj := Value;
end;

procedure TStringObject.SetTag(const Value: Integer);
begin
```

```
  FTag := Value;
end;

procedure TStringObject.SetValue(const Value: TString);
begin
  FValue := Value;
end;

{ TStringObjectArray }

function TStringObjectArray.Add: Integer;
begin
 Result:=inherited Add;
 CreateItem(Result);
end;

function TStringObjectArray.AddItem(const Item): Integer;
begin
 Result:=Add;
end;

constructor TStringObjectArray.Create;
begin
 inherited Create(0, SizeOf(TStringObject));
end;

procedure TStringObjectArray.CreateItem(Index: Integer);
var
 P: ^TStringObject;
begin
 P:=GetItemPtr(Index);
 P^:=TStringObject.Create;
end;

procedure TStringObjectArray.Delete(Index: Integer);
begin
 FreeItem(Index);
 inherited;
end;

procedure TStringObjectArray.DeleteItem(Index: Integer; out Item);
begin
 Delete(Index);
end;

destructor TStringObjectArray.Destroy;
begin
 ForEach(Integer(Self), @TStringObjectArray.FreeObject);
 inherited;
end;

procedure TStringObjectArray.FreeItem(Index: Integer);
var
 P: ^TStringObject;
begin
 P:=GetItemPtr(Index);
 FreeAndNil(P^);
end;

function TStringObjectArray.FreeObject(Index: Integer;
  var Obj: TStringObject): Integer;
begin
 FreeAndNil(Obj);
 Result:=0;
end;

function TStringObjectArray.GetData(Index: Integer): Pointer;
begin
 Result:=GetObject(Index).Data;
```

```
end;

function TStringObjectArray.GetObject(Index: Integer): TStringObject;
begin
 GetItem(Index, Result);
end;

function TStringObjectArray.GetRefObj(Index: Integer): TObject;
begin
 Result:=GetObject(Index).RefObj;
end;

function TStringObjectArray.GetTag(Index: Integer): Integer;
begin
 Result:=GetObject(Index).Tag;
end;

function TStringObjectArray.GetValue(Index: Integer): TString;
begin
 Result:=GetObject(Index).Value;
end;

procedure TStringObjectArray.Insert(Index: Integer);
begin
 inherited;
 CreateItem(Index);
end;

procedure TStringObjectArray.InsertItem(Index: Integer; const Item);
begin
 Insert(Index);
end;

procedure TStringObjectArray.SetCount(const NewCount: Cardinal);
var
 i, OldCount: Integer;
begin
 OldCount:=Count;
 if NewCount > Count then begin
  inherited SetCount(NewCount);
  for i:=OldCount to NewCount - 1 do CreateItem(i);
 end else if NewCount < Count then begin
  for i:=NewCount to OldCount - 1 do FreeItem(i);
  inherited SetCount(NewCount);
 end;
end;

procedure TStringObjectArray.SetData(Index: Integer; const Value: Pointer);
begin
 GetObject(Index).Data:=Value;
end;

procedure TStringObjectArray.SetRefObj(Index: Integer;
 const Value: TObject);
begin
 GetObject(Index).RefObj:=Value;
end;

procedure TStringObjectArray.SetTag(Index: Integer; const Value: Integer);
begin
 GetObject(Index).Tag:=Value;
end;

procedure TStringObjectArray.SetValue(Index: Integer; const Value: TString);
begin
 GetObject(Index).Value:=Value;
end;

{ TStringObjectList }
```

```
function TStringObjectList.Add(const S: string): Integer;
begin
 Result:=FArray.Add;
 FArray.Value[Result]:=S;
end;

procedure TStringObjectList.Clear;
begin
 FArray.Count:=0;
end;

constructor TStringObjectList.Create;
begin
 inherited Create;
 FArray:=TStringObjectArray.Create;
end;

procedure TStringObjectList.Delete(Index: Integer);
begin
 FArray.Delete(Index);
end;

destructor TStringObjectList.Destroy;
begin
 FArray.Free;
 inherited;
end;

procedure TStringObjectList.Exchange(Index1, Index2: Integer);
begin
 FArray.Swap(Index1, Index2);
end;

function TStringObjectList.Get(Index: Integer): string;
begin
 Result:=FArray.Value[Index];
end;

function TStringObjectList.GetCount: Integer;
begin
 Result:=FArray.Count;
end;

function TStringObjectList.GetData(Index: Integer): Pointer;
begin
 Result:=FArray.Data[Index];
end;

function TStringObjectList.GetObject(Index: Integer): TObject;
begin
 Result:=FArray.RefObj[Index];
end;

function TStringObjectList.GetTag(Index: Integer): Integer;
begin
 Result:=FArray.Tag[Index];
end;

procedure TStringObjectList.Insert(Index: Integer; const S: string);
begin
 FArray.Insert(Index);
 FArray.Value[Index]:=S;
end;

procedure TStringObjectList.Put(Index: Integer; const S: string);
begin
 FArray.Value[Index]:=S;
end;
```

```
procedure TStringObjectList.PutObject(Index: Integer; AObject: TObject);
begin
 FArray.RefObj[Index]:=AObject;
end;

procedure TStringObjectList.SetData(Index: Integer; const Value: Pointer);
begin
 FArray.Data[Index]:=Value;
end;

procedure TStringObjectList.SetTag(Index: Integer; const Value: Integer);
begin
 FArray.Tag[Index]:=Value;
end;

{ TNetworkNeighborhood }

function TNetworkNeighborhood.Add(const S: string): Integer;
begin
 Result:=inherited Add(S);
 Objects[Result]:=TNetworkWorkgroup.Create;
end;

procedure TNetworkNeighborhood.Clear;
begin
 FArray.ForEach(Integer(Self), @TNetworkNeighborhood.FreeRefObj);
 inherited;
end;

function TNetworkNeighborhood.CopyPIDL(IDList: PItemIDList): PItemIDList;
var
 Size: Integer;
begin
 Size := GetPIDLSize(IDList);
 Result := CreatePIDL(Size);
 if Assigned(Result) then CopyMemory(Result, IDList, Size);
end;

constructor TNetworkNeighborhood.Create;
begin
 inherited Create;
 Refresh;
end;

function TNetworkNeighborhood.CreatePIDL(Size: Integer): PItemIDList;
var
 Malloc: IMalloc;
 HR: HResult;
begin
 Result := nil;
 HR := SHGetMalloc(Malloc);
 if Failed(HR) then  Exit;
 try
  Result := Malloc.Alloc(Size);
  if Assigned(Result) then FillChar(Result^, Size, 0);
 finally
 end;
end;

procedure TNetworkNeighborhood.Delete(Index: Integer);
begin
end;

procedure TNetworkNeighborhood.DisposePIDL(ID: PItemIDList);
var
 Malloc: IMalloc;
begin
 if ID = nil then Exit;
```

```
 OLECheck(SHGetMalloc(Malloc));
 Malloc.Free(ID);
end;

class function TNetworkNeighborhood.EnumObjects(
  ShellFolder: IShellFolder): IEnumIDList;
const
 Flags = SHCONTF_FOLDERS or SHCONTF_NONFOLDERS or SHCONTF_INCLUDEHIDDEN;
begin
 ShellFolder.EnumObjects(0, Flags, Result);
end;

function TNetworkNeighborhood.FindComputer(Name: TString): TString;
var
 i, j: Integer;
 List: TNetworkWorkgroup;
 S: TString;
begin
 Result:=' ' ;
 try
  for i:=0 to Count - 1 do begin
   List:=Objects[i] as TNetworkWorkgroup;
   for j:=0 to List.Count - 1 do begin
    S:=List[j];
    CleanUp(S);
    if EqualText(Name, S) then begin
     Result:=Strings[i];
     raise ComputerFound.Create(' ' );
    end;
   end;
  end;
 except
  if not (ExceptObject is ComputerFound) then raise;
 end;
end;

function TNetworkNeighborhood.FreeRefObj(Index: Integer;
  var Obj: TStringObject): Integer;
begin
 FreeAndNil(Obj.FRefObj);
 Result:=0;
end;

class function TNetworkNeighborhood.GetDisplayName(ShellFolder: IShellFolder;
  PIDL: PItemIDList): TString;
var
 StrRet: TStrRet;
 P: PChar;
begin
 Result := ' ' ;
 ShellFolder.GetDisplayNameOf(PIDL, SHGDN_NORMAL, StrRet);
 case StrRet.uType of
  STRRET_CSTR: SetString(Result, StrRet.cStr, lStrLen(StrRet.cStr));
  STRRET_OFFSET: begin
   P := @PIDL.mkid.abID[StrRet.uOffset - SizeOf(PIDL.mkid.cb)];
   SetString(Result, P, PIDL.mkid.cb - StrRet.uOffset);
  end;
  STRRET_WSTR: Result := StrRet.pOleStr;
 end;
 CleanUp(Result, True);
end;

function TNetworkNeighborhood.GetPIDLSize(IDList: PItemIDList): Integer;
begin
 Result := 0;
 if Assigned(IDList) then begin
  Result := SizeOf(IDList^.mkid.cb);
  while IDList^.mkid.cb <> 0 do begin
   Result := Result + IDList^.mkid.cb;
```

```
   IDList := NextPIDL(IDList);
  end;
 end;
end;

function TNetworkNeighborhood.GetPrevPIDL(PIDL: PItemIDList): PItemIDList;
var
 Temp: PItemIDList;
begin
 Temp := CopyPIDL(PIDL);
 if Assigned(Temp) then StripLastID(Temp);
 if Temp.mkid.cb <> 0 then Result:=Temp else Result:=nil;
end;

function TNetworkNeighborhood.GetWorkgroup(Name: TString): TNetworkWorkgroup;
var
 Index: Integer;
begin
 Index:=IndexOf(Name);
 if Index<>-1 then Result:=Objects[Index] as TNetworkWorkgroup else Result:=nil;
end;

procedure TNetworkNeighborhood.Insert(Index: Integer; const S: string);
begin
end;

procedure TNetworkNeighborhood.ListComputers(Strings: TStrings);
var
 i, j: integer;
 L: TNetworkWorkgroup;
 S: TString;
begin
 Strings.BeginUpdate;
 try
  Strings.Clear;
  for i:=0 to Count - 1 do begin
   L:=Objects[i] as TNetworkWorkgroup;
   for j:=0 to L.Count - 1 do begin
    S:=L[j];
    CleanUp(S);
    Strings.Add(S);
   end;
  end;
 finally
  Strings.EndUpdate;
 end;
end;

procedure TNetworkNeighborhood.ListNetwork(Strings: TStrings);
var
 List: TStringList;
 i: Integer;
begin
 List:=TStringList.Create;
 try
  List.AddStrings(Self);
  for i:=0 to List.Count - 1 do List.Objects[i]:=TObject(1);
  for i:=0 to Count - 1 do begin
   List.AddStrings(Objects[i] as TStrings);
  end;
  for i:=Count to List.Count - 1 do List.Objects[i]:=nil;
  List.Sort;
  Strings.Assign(List);
 finally
  List.Free;
 end;
end;

function TNetworkNeighborhood.NextPIDL(IDList: PItemIDList): PItemIDList;
```

```
begin
 Result := IDList;
 Inc(PChar(Result), IDList^.mkid.cb);
end;

function TNetworkNeighborhood.OriginFolder: IShellFolder;
var
 Desktop: IShellFolder;
 S: TString;
 P: PWideChar;
 Len, Flags: LongWord;
 Machine, Workgroup, Network: PItemIDList;
begin
 S:=' \\' +GetComputerName;
 Len:=Length(S);
 P:=StringToOleStr(S);
 Flags:=0;
 SHGetDesktopFolder(Desktop);
 Desktop.ParseDisplayName(0, nil, P, Len, Machine, Flags);
 Workgroup:=GetPrevPIDL(Machine);
 try
   Network:=GetPrevPIDL(Workgroup);
  try
   Desktop.BindToObject(Network, nil, IShellFolder, Pointer(Result));
  finally
   DisposePIDL(Network);
  end;
 finally
  DisposePIDL(Workgroup);
 end;
end;

function TNetworkNeighborhood.OriginFolderNT: IShellFolder;
var
 Desktop: IShellFolder;
 S: TString; W: WideString; P: PWideChar;
 Len, Flags: LongWord;
 Machine, Workgroup, Network: PItemIDList;
 NetShell: IShellFolder;
 Enum: IEnumIDList;
 ID: PItemIDList;
begin
 S:=' \\' +GetComputerName;
 Len:=Length(S);
 W:=S; P:=PWideChar(W);
 SHGetDesktopFolder(Desktop);
 Desktop.ParseDisplayName(0, nil, P, Len, Machine, Flags);
 Workgroup:=GetPrevPIDL(Machine);
 Network:=GetPrevPIDL(Workgroup);
 Desktop.BindToObject(Network, nil, IShellFolder, NetShell);
 Enum:=EnumObjects(NetShell);
 Enum.Next(1, ID, Flags);
 NetShell.BindToObject(ID, nil, IShellFolder, Pointer(Result));
 DisposePIDL(Network);
 DisposePIDL(Workgroup);
end;

class procedure TNetworkNeighborhood.ParseFolder(Folder: IShellFolder;
  Items: TStringObjectList; StorePIDLs: Boolean);
var
 ID: PItemiDList;
 EnumList: IEnumIDList;
 NumIDs: LongWord;
 S: TString;
 Index: Integer;
begin
 Items.BeginUpdate;
 try
  Items.Clear;
```

```
  EnumList:=EnumObjects(Folder);
  if Assigned(EnumList) then while EnumList.Next(1, ID, NumIDs) = S_OK do begin
   S:=GetDisplayName(Folder, ID);
   Index:=Items.Add(S);
   if StorePIDLs then Items.Data[Index]:=ID;
  end;
 finally
  Items.EndUpdate;
 end;
end;

class procedure TNetworkNeighborhood.ParseFolderEx(Folder: IShellFolder;
  Items: TStrings);
var
 ID: PItemiDList;
 EnumList: IEnumIDList;
 NumIDs: LongWord;
 S: TString;
begin
 Items.BeginUpdate;
 try
  Items.Clear;
  EnumList:=EnumObjects(Folder);
  if Assigned(EnumList) then while EnumList.Next(1, ID, NumIDs) = S_OK do begin
   S:=GetDisplayName(Folder, ID);
   Items.Add(S);
  end;
 finally
  Items.EndUpdate;
 end;
end;

procedure TNetworkNeighborhood.Refresh;
var
 Network: IShellFolder;
 Workgroup: IShellFolder;
 i: Integer;
begin
 try
  if WinNT and (not Win2K) then Network:=OriginFolderNT else
   Network:=OriginFolder;
  ParseFolder(Network, Self, True);
  for i:=0 to Count - 1 do begin
   Network.BindToObject(PItemIDList(Data[i]), nil, IShellFolder, Workgroup);
   ParseFolder(Workgroup, Objects[i] as TStringObjectList, False);
   Workgroup:=nil;
  end;
 except
  raise ECannotFindNetwork.Create(SCannotFindNetwork);
 end;
end;

procedure TNetworkNeighborhood.StripLastID(IDList: PItemIDList);
var
 MarkerID: PItemIDList;
begin
 MarkerID := IDList;
 if Assigned(IDList) then begin
  while IDList.mkid.cb <> 0 do begin
   MarkerID := IDList;
   IDList := NextPIDL(IDList);
  end;
  MarkerID.mkid.cb := 0;
 end;
end;


procedure GetIPAddresses(Network: TNetworkNeighborhood; List: TStrings);
var
```

```
 Error: DWORD;
 HostEntry: PHostEnt;
 Data: WSAData;
 Address: In_Addr;
 i: Integer;
 TmpList: TStringList;
 S: TString;
begin
{ List.BeginUpdate;
 try}
  List.Clear;
  Error:=WSAStartup(MakeWord(1, 1), Data);
  if Error = 0 then begin
   TmpList:=TStringList.Create;
   try
    Network.ListComputers(TmpList);
    for i:=0 to TmpList.Count - 1 do begin
     HostEntry:=gethostbyname(PChar(TmpList[i]));
     Error:=GetLastError;
     if Error <> 0 then S:=‘ Unknown’  else begin
      Address:=PInAddr(HostEntry^.h_addr_list^)^;
      S:=inet_ntoa(Address);
     end;
     List.Add(Format(‘ %s [%s]’ , [TmpList[i], S]));
    end;
   finally
    TmpList.Free;
   end;
  end else begin
   List.Add(‘ Error’ );
  end;
{ finally
  List.EndUpdate;
 end;}
end;


function GetShellFolder(ComputerName: TString): IShellFolder;
var
 S: TString;
 W: WideString;
 P: PWideChar;
 Desktop: IShellFolder;
 Len, Flags: LongWord;
 Machine: PItemIDList;
begin
 S:=ComputerName;
 if Pos(‘ \\’ , S) <> 1 then S:=‘ \\’ +S;
 Len:=Length(S);
 W:=S;
 P:=@W[1];
 SHGetDesktopFolder(Desktop);
 Desktop.ParseDisplayName(0, nil, P, Len, Machine, Flags);
 Desktop.BindToObject(Machine, nil, IShellFolder, Pointer(Result));
end;

function EnumSharedResources(ComputerName: TString; List: TStrings): Boolean;
var
 ShellFolder: IShellFolder;
begin
 ShellFolder:=GetShellFolder(ComputerName);
 Result:=Assigned(ShellFolder);
 if Result then TNetworkNeighborhood.ParseFolderEx(ShellFolder, List);
end;



function GetIPAddress(NetworkName: TString): TString;
var
```

```
 Error: DWORD;
 HostEntry: PHostEnt;
 Data: WSAData;
 Address: In_Addr;
begin
 Error:=WSAStartup(MakeWord(1, 1), Data);
 if Error = 0 then begin
  HostEntry:=gethostbyname(PChar(NetworkName));
  Error:=GetLastError();
  if Error = 0 then begin
   Address:=PInAddr(HostEntry^.h_addr_list^)^;
   Result:=inet_ntoa(Address);
  end else begin
   Result:=‘ Unknown’ ;
  end;
 end else begin
  Result:=‘ Error’ ;
 end;
 WSACleanup();
end;

end.
```

Файл About.pas - довідка

```
unit About;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls;

type
  TForm2 = class(TForm)
    Image1: TImage;
    Memo1: TMemo;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.Button1Click(Sender: TObject);
begin
Form2.Close;
end;

end.
```