МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Кафедра кібербезпеки та програмного забезпечення

На правах рукопису

Соловйов Роман Андрійович

Програмне забезпечення системи оцінювання якості каналів зв'язку MIMO для 5G

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

Соловйову Роману Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи оцінювання якості каналів зв'язку MIMO для 5G*

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 оцінювання якості каналів зв'язку MIMO для 5G*

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

Соловйов Р.А. Програмне забезпечення системи оцінювання якості каналів зв'язку MIMO для 5G. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній кваліфікаційній бакалаврській розроблено програмне забезпечення, яке призначено для системи оцінювання якості каналів зв'язку MIMO для 5G.

Метою розробки є програмне забезпечення системи оцінювання якості каналів зв'язку MIMO для 5G.

Результат роботи – програмна реалізація системи оцінювання якості каналів зв'язку MIMO для 5G.

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 Delphi 10.4 Sydney.

Ключові слова: комп'ютерна інженерія, MIMO, 5G

ABSTRACT

Soloviov R.A. MIMO 5G channel quality assessment system software. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

This undergraduate qualification has developed software designed for the MIMO communication channel quality assessment system for 5G.

The purpose of the development is the software of the MIMO communication quality assessment system for 5G.

The result is a software implementation of the MIMO communication channel quality assessment system for 5G.

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 Delphi 10.4 Sydney.

Keywords: computer engineering, MIMO, 5G

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

5G	–	Мобільний зв'язок 5-го покоління
АС	–	Абонентська станція
АЦП	–	Аналого-цифрові перетворювачі
БС	–	Базова станція
ЧР	–	Частотний ресурс
ЦАП	–	Цифро-аналогові перетворювачі
CSI-RS	–	Channel state information – reference signal
DM-RS	–	Demodulation reference signal
FDD	–	Frequency division duplex
MBS	–	Macro Base Station
MIMO	–	Multiple Input Multiple Output, використання декількох передавальних і прийомних антен
MISO	–	Multiple Input Single Output, рознесена передача
MU-MIMO	–	Multi User MIMO
PMI	–	Precoding matrix indicator
SIMO	–	Single Input Multiple Output, рознесене приймання
SINR	–	Signal to Interference plus Noise Ratio
SISO	–	Single Input Single Output
SU-MIMO	–	Single User MIMO
TDD	–	Time division duplex
WISP	–	Бездротові мережі широкосмугового доступу

ВСТУП

Актуальність теми. Вимоги до пропускної здатності мобільних мереж дуже високі й, при цьому, вони постійно ростуть. Очевидні варіанти збільшення пропускної здатності – збільшення ширини каналу й використання модуляцій більш високого порядку, не дозволяють повністю розв'язати завдання забезпечення високої пропускної здатності. Частотний діапазон все-таки обмежений. А використання модуляції більш високого порядку має на увазі підвищення SINR (Signal to Interference plus Noise Ratio), що теж має своя межу. Ще одним способом збільшення пропускної здатності бездротових систем є використання декількох передавальних і прийомних антен (MIMO – Multiple Input Multiple Output) і спеціальна обробка сигналу в цьому випадку.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи оцінювання якості каналів зв’язку MIMO для 5G.

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

– Огляд існуючих систем оцінювання якості каналів зв’язку MIMO для 5G.

– Дослідження системи оцінювання якості каналів зв’язку MIMO для 5G.

– Програмна реалізація системи оцінювання якості каналів зв’язку MIMO для 5G.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі оцінювання якості каналів зв’язку MIMO для 5G.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи оцінювання якості каналів зв’язку MIMO для 5G, є актуальною задачею, яка потребує вирішення у даній кваліфікаційній бакалаврській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Приведемо класифікацію варіантів MIMO і їх короткий опис.

Класична система (SISO – Single Input Single Output)

Для початку розглянемо варіанти MIMO, які можуть бути використані для передачі даних одному користувачеві. Перший класичний і найпростіший варіант використання однієї передавальної й одному прийомної антени зображений на рисунку нижче. Така система з погляду термінології MIMO називається SISO – Single Input Single Output.

Пропускну здатність такої системи можна розрахувати, використовуючи формулу Шеннона:

$$C = B \log_2(1 + S/N),$$

де:

– C – пропускна здатність каналу;

– B – ширина каналу;

– S/N – співвідношення сигнал/шум.

Рознесене приймання (Rx Diversity, SIMO – Single Input Multiple Output)

Рознесене приймання (Rx Diversity) – це випадок використання більшої кількості антен на прийомній стороні, чому на передавальній. З погляду MIMO така система називається SIMO – Single Input Multiple Output. Найпростіший випадок такої системи антена, що коли передає, одна, а приймальних дві, називається SIMO 1x2.

Представлений варіант не вимагає спеціальної підготовки сигналу при передачі, тому його досить просто реалізувати на практиці. При використанні рознесеного приймання збільшення пропускної здатності не відбувається. Однак,

підвищується надійність передачі. У випадку із зображеною вище системою на прийомній стороні буде два сигнали, і існують різні способи їх обробки. Наприклад, може вибиратися сигнал з найкращим співвідношенням сигнал/шум. Такий метод називається switched diversity. Або сигнали можуть складатися, що дозволяє підвищити співвідношення сигнал/шум. І такий метод називається MRC – Maximum Ratio Combining.

Рознесена передача (Tx Diversity, MISO – Multiple Input Single Output)

Рознесена передача (Tx Diversity) – це випадок використання більшої кількості антен на передавальній стороні, чому на приймальні. З погляду MIMO така система називається MISO – Multiple Input Single Output. Найпростіший випадок такої системи антен, що коли передають, дві, а приймальня одна, називається MISO 2x1.

Як і SIMO, MISO не дозволяє збільшити пропускну здатність каналу, але підвищує надійність передачі. У той же час, використання MISO дозволяє перенести необхідну додаткову обробку сигналу із прийомної сторони (мобільної станції) на передавальну (базову станцію). Для формування надійного сигналу використовується просторово-тимчасове кодування. У цьому випадку копія сигналу передається не тільки з іншої антени, але й іншим часом. Також може використовуватися просторово-частотне кодування.

Просторове ущільнення (Spatial Multiplexing, MIMO – Multiple Input Multiple Output)

Просторове ущільнення (Spatial Multiplexing) – це випадок використання декількох антен на передавальній стороні й декількох антен на приймальні. На відміну від попередніх варіантів – MISO і SIMO, описаних вище, даний варіант спрямований не на підвищення надійності передачі, а на збільшення швидкості передачі. Тому MIMO використовується для передачі даних мобільним станціям, які перебувають у гарних радіоумовах. У той час, як варіанти MISO і SIMO використовуються для передачі даних мобільним станціям, які перебувають у більш поганих радіоумовах. Для того, щоб підвищити швидкість передачі даних у

випадку з MIMO вхідний потік даних розбивають на кілька потоків, кожний з яких незалежно передається з окремої антени.

Через те, що використовується загальний канал, кожна антена на приймачі одержує сигнал не тільки призначений для неї, але й усі сигнали призначені іншим антенам. Якщо відома матриця передачі, то вплив сигналів, призначених для інших антен, можна обчислити й мінімізувати.

Кількість незалежних потоків даних, які можуть одночасно передаватися, залежить від кількості використовуваних антен. Якщо кількість передавальних і прийомних антен однаково, то кількість незалежних потоків даних рівно або менше кількості антен. Наприклад, у випадку MIMO 4x4 кількість незалежних потоків даних може бути 4 або менше. Якщо ж кількість передавальних і прийомних антен не однаково, то кількість незалежних потоків даних дорівнює мінімальній кількості антен або менше. Наприклад у випадку MIMO 4x2 кількість незалежних потоків даних може бути 2 або менше.

Для обчислення максимальної пропускної здатності у випадку використання MIMO застосовується наступна формула:

$$C = M\,B\,\log_2(1 + S/N),$$

де:

– C – пропускна здатність каналу;

– M – кількість незалежних потоків даних;

– B – ширина каналу;

– S/N – співвідношення сигнал/шум.

Залежно від кількості користувачів, яким одночасно здійснюється передача даних, можна виділити наступні варіанти. Single User MIMO (SU-MIMO) – коли технологія MIMO використовується для передачі даних одному користувачеві, тобто всі потоки даних адресовані тому самому користувачеві. І Multi User MIMO (MU-MIMO) – коли технологія MIMO використовується для передачі даних декільком користувачам одночасно в тих самих ресурсних блоках,

тобто коли незалежні потоки даних адресовані різним користувачам. Нижче на рисунку приводиться приклад MU-MIMO для випадку із двома користувачами.

1.2 Область застосування

Буде чи 5G тільки еволюцією 4G, або новітні технології приведуть до потрясінь, які зажадають масового переосмислення вкорінених принципів мобільних мереж?

Ми розглядаємо вплив нових технологій, використовуючи модель Хендерсона-Кларка (модель класифікації інновацій), у такий спосіб:

– Еволюція дизайну. Незначні зміни як вузлового, так і архітектурного рівнів (наприклад, уведення класифікаторів і сигналізації, підтримка більшого числа антен).

– Компонентні зміни. Проривні зміни в дизайні класу мережних вузлів (наприклад, уведення нових хвильових форм).

– Архітектурні зміни. Проривні зміни в системній архітектурі (наприклад, уведення нових типів вузлів або введення нових функцій у вже існуючі).

– Радикальні зміни. Проривні зміни, що виявляють вплив на вузли й рівні архітектури.

Орієнтир на проривні (компонентні, архітектурні, радикальні) технології обумовлений, за нашим переконанням, надзвичайно високою сукупною швидкістю передачі даних і низкою затримкою, що вимагаються для 5G, що не може бути досягнуте лише простою еволюцією статус-кво.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи оцінювання якості каналів зв’язку MIMO для 5G , 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

Sector MIMO 5G-120 (SECM5120)

Антени RF elements Mimo Sector – установили новий стандарт по співвідношенню ціна/якість, сумісності з іншими пристроями, простоті використання й стійкості до впливу навколишнього середовища. Секторні антени RF elements підтримують технологію 2x2 MIMO, тобто вони працюють у подвійній поляризації (вертикальна + горизонтальна) Антена Sector MIMO 5G-120 має систему кріплення яка дозволяє робити швидку установку й підключення точок доступу до антени. Ця система повністю сумісна з корпусами StatioNBox S від RF elements, які призначені для пристроїв Routerboard серії 411/711 компанії Mikrotik, і точками доступу Rocket M5, Rocket M5 GPS виробника Ubiquiti Networks. Секторна антена Sector MIMO 5G-1200 відмінно підійде для організації базових станцій будь-якого рівня, а сумісність із більшою кількістю бездротового устаткування робить установку більш гнучкої й оптимальної для кожного користувача. Секторна антена Sector MIMO 5G-120 має естетичний дизайн і відмінну функціональну конструкцію. Антена має плавні вигини й майже органічні форми. У виготовленні використовувалися тільки тверді і якісні антикорозійні матеріали, ABS пластик захищає антену від впливу ультрафіолетового випромінювання, кріплення антени виготовлені з алюмінію, литого під тиском. Мала вага антени (1,4 кг) знижує загальне навантаження на щоглу, що уможливлює установку антени на високі щогли. На задній панелі антени встановлюється спеціальний ковпачок для захисту портів точки доступу від впливу сонячних променів, дощу, снігу, льоду або злих птахів які можуть ушкодити сполучні пігтейли між точкою доступу й антеною Sector MIMO 5G-

120. Простий й ефективний застосунок. Інтеграція GPS антени Для пристроїв які використовують GPS синхронізацію, таких як Rocket M5 GPS, в антені Sector MIMO 5G-120 передбачений спеціальний відсік для GPS-антен. При установці, GPS антена повністю міститься в корпус антени, включаючи й пігтейл, у результаті чого всі надійно захищене й немає кабелів, що стирчать.

Sector MIMO 5G-90 (SECM590)

Антени RF elements Mimo Sector – установили новий стандарт по співвідношенню ціна/якість, сумісності з іншими пристроями, простоті використання й стійкості до впливу навколишнього середовища. Секторні антени RF elements підтримують технологію 2x2 MIMO, тобто вони працюють у подвійній поляризації (вертикальна + горизонтальна) Антена Sector MIMO 5G-90 має систему кріплення яка дозволяє робити швидку установку й підключення точок доступу до антени. Ця система повністю сумісна з корпусами StatioNBox S від RF elements, які призначені для пристроїв Routerboard серії 411/711 компанії Mikrotik, і точками доступу Rocket M5, Rocket M5 GPS виробника Ubiquiti Networks. Секторна антена Sector MIMO 5G-90 відмінно підійде для організації базових станцій будь-якого рівня, а сумісність із більшою кількістю бездротового устаткування робить установку більш гнучкою й оптимальною для кожного користувача. Секторна антена Sector MIMO 5G-90 має естетичний дизайн і відмінну функціональну конструкцію. Антена має плавні вигини й майже органічні форми. У виготовленні використовувалися тільки тверді і якісні антикорозійні матеріали, ABS пластик захищає антену від впливу ультрафіолетового випромінювання, кріплення антени виготовлені з алюмінію, литого під тиском . Мала вага антени (1,4 кг) знижує загальне навантаження на щоглу, що уможливлює установку антени на високі щогли. На задній панелі антени встановлюється спеціальний ковпачок для захисту портів точки доступу від впливу сонячних променів, дощу, снігу, льоду або злих птахів які можуть ушкодити сполучні пігтейли між точкою доступу й антеною Sector MIMO 5G-90. Простої й ефективний застосунок. Інтеграція GPS антени для пристроїв які

використовують GPS синхронізацію, таких як Rocket M5 GPS, в антені Sector MIMO 5G-90 передбачений спеціальний відсік для GPS-антен. При установці, GPS антена повністю міститься в корпус антени, включаючи й пігтейл, у результаті чого всі надійно захищене й немає кабелів, що стирчать.

ePMP 1000 5 Ghz Sector 90 Antenna

Інноваційна лінійка бездротового устаткування для побудови мереж по типу точка-багатоточка – ePMP, від компанії Cambium Networks, представлена в нашому магазині, є досить гнучким інструментом, і може стати прекрасним розв'язком для більшості завдань, що коштують перед сучасними операторами бездротових мереж широкосмугового доступу (WISP). І одним з невід'ємних компонентів, будь-якої системи, призначеної для такого роду мереж, є секторні антени. ePMP 5 Ghz Sector Antenna – спеціально розроблена компанією Cambium, для роботи разом з базовою станцією ePMP 1000 GPS Sync у бездротових мережах широкосмугового доступу в частотному діапазоні 5150 – 5850Mhz. Антена має високий коефіцієнт підсилення в 16.5dbi при ширині променя сектору в 90º. Підтримує дві поляризації (горизонтальна й вертикальна), що забезпечує підтримку технології 2x2 MIMO при роботі з базовими станціями ePMP 1000. І спроектована таким чином, що при роботі в багатосекторної базової станції ePMP з режимом GPS синхронізації, забезпечує повторне використання радіочастотних каналів і вкрай високий коефіцієнт придушення випромінювання в протилежну сторону, який відповідає >30db. Усе це, дозволяє не тільки максимально “ощадливо” використовувати частотні ресурси, але й розвертати бездротові мережі в місцях з високою їхньою щільністю. Для підключення до базової станції, використовуються два RP -SMA рознімання. А спеціальне кріплення, дозволяє надійно зафіксувати пристрою ePMP 1000 GPS Sync, безпосередньо на задній стороні антени, забезпечивши їх захист. Крім того, антени ePMP 5 Ghz Sector Antenna, виконані з дотриманням усіх вимог до устаткування операторського класу. Виконані з якісних матеріалів, здатних протистояти

несприятливим погодним і кліматичним умовам, і мають високий рівень вітрового навантаження, до 190км/ч.

ePMP 1000 5 Ghz Sector 120 Antenna

Інноваційна лінійка бездротового устаткування для побудови мереж по типу точка-багатоточка – ePMP, від компанії Cambium Networks, представлена в нашому магазині, є досить гнучким інструментом, і може стати прекрасним розв'язком для більшості завдань, що коштують перед сучасними операторами бездротових мереж широкосмугового доступу (WISP). І одним з невід'ємних компонентів, будь-якої системи, призначеної для такого роду мереж, є секторні антени. ePMP 5 Ghz Sector Antenna – спеціально розроблена компанією Cambium, для роботи разом з базовою станцією ePMP 1000 GPS Sync у бездротових мережах широкосмугового доступу в частотному діапазоні 5150 – 5850Mhz. Антена має високий коефіцієнт підсилення в 15.5dbi при ширині променя сектору в 120º. Підтримує дві поляризації (горизонтальна й вертикальна), що забезпечує підтримку технології 2x2 MIMO при роботі з базовими станціями ePMP 1000. І спроектована таким чином, що при роботі в багатосекторній базовій станції ePMP з режимом GPS синхронізації, забезпечує повторне використання радіочастотних каналів і вкрай високий коефіцієнт придушення випромінювання в протилежну сторону, який відповідає >30db. Усе це, дозволяє не тільки максимально “ощадливо” використовувати частотні ресурси, але й розвертати бездротові мережі в місцях з високою їхньою щільністю. Для підключення до базової станції, використовуються два RP-SMA рознімання. А спеціальне кріплення, дозволяє надійно зафіксувати пристрою ePMP 1000 GPS Sync, безпосередньо на задній стороні антени, забезпечивши їх захист. Крім того, антени ePMP 5 Ghz Sector Antenna, виконані з дотриманням усіх вимог до устаткування операторського класу. Виконані з якісних матеріалів, здатних протистояти несприятливим погодним і кліматичним умовам, і мають високий рівень вітрового навантаження, до 190км/ч.

Нові технології зв'язку для розумних, підключених і безпечних пристроїв

5G – стандарт стільникової мережі п'ятого покоління – забезпечує винятково швидкий зв'язок, наднадійні канали зв'язку з низьким рівнем затримок, взаємодію в реальному часі, а також підтримку величезної кількості підключених пристроїв на невеликих територіях. 5G не тільки забезпечує швидкий доступ до Інтернету для більшого числа користувачів, але й дозволяє використовувати нові технології й бізнес-процеси в різних галузях: від сучасного автоматизованого виробництва до безпечних автономних автомобілів і дистанційної хірургії.

Виробники пристроїв і постачальники мережної інфраструктури повинні забезпечувати очікувану високу продуктивність за короткий час, з мінімальними витратами й з дотриманням усіх нормативних вимог і стандартів безпеки. Компанії, що інвестують у ці галузі, зустрічаються зі значними технологічними проблемами, багато з яких залишаються невирішеними. Для повсюдного поширення каналів зв'язки з діапазоном частот менш 6 ГГц і міліметровим діапазоном потрібне масове використання технології MIMO (багатоканальний вхід, багатоканальний вихід) і технології антенних ґрат у бездротових точках доступу (наприклад, у мілкостільникових базових станціях) і в користувацьких пристроях, включаючи смартфони.

Підтримка стандартів 5G поряд з величезною кількістю вже існуючих пристроїв створює величезне навантаження на інженерів, які повинні розробляти інноваційні високопродуктивні системи, що не викликають проблем електромагнітної сумісності або електромагнітних перешкод (EMC/EMI), таких як глушіння сигналів інших систем зв'язки. Додаткові фактори, такі як розробка прийнятних характеристик терморегуляції для пристроїв, оснащених більшою кількістю компонентів, або підтримка більш точних механічних допусків для компонентів, що працюють у міліметровому діапазоні, збільшують потребу в комплексному мультифізичному і багатопрофільному проектуванні.

Точне імітаційне моделювання всіх фізичних аспектів на всіх етапах процесу проектування, від первісного вивчення концепції до віртуального тестування на відповідність нормативним вимогам, має дуже велике значення. Великий портфель технологій моделювання SIMULIA від Dassault Systèmes містить у собі механічний аналіз, структурний аналіз і аналіз напруг за допомогою Abaqus, електромагнітне моделювання за допомогою CST Studio Suite, а також інші інструменти для температурного, втомленостного й системного моделювання. У комбінації з перевагами платформи 3DEXPERIENCE ці технології дозволяють випускати конкурентоспроможні продукти в строк і в рамках бюджету, незважаючи на зростаючі складності, пов'язані з підтримкою стандарту 5G.

Abaqus

У цей час технічні фахівці нерідко використовують вузькоспеціалізовані інструменти різних постачальників для моделювання різних атрибутів проектованих продуктів. Використання різних програмних продуктів знижує ефективність і збільшує витрати. SIMULIA надає масштабований пакет уніфікованих застосунків для аналізу, за допомогою яких користувачі (незалежно від галузі й досвіду в моделюванні) можуть спільно працювати, безперешкодно обмінюватися даними моделювання й застосовувати перевірені методи без шкоди для вірогідності інформації.

Пакет продуктів Abaqus Unified FEA пропонує потужні комплексні розв'язки не тільки для роботи з поточними завданнями, але й для усунення складних інженерних проблем у самих різних галузях. Так, наприклад, в автомобільній промисловості робочі технічні групи на основі структури даних моделювання й інтегрованої вирішальної технології можуть здійснювати розрахунки повного навантаження автомобіля, динамічних вібрацій, багатомодульних систем, наслідків удару/аварії, проводити нелінійний статичний аналіз і визначати тепловий і акустико-структурний зв'язки. Провідні галузеві компанії використовують пакет Abaqus Unified FEA для консолідації процесів і

інструментів, скорочення витрат, підвищення ефективності й одержання конкурентних переваг.

CST STUDIO SUITE

CST Studio Suite® – це високопродуктивний пакет програмного забезпечення для ЕМ-аналізу в 3D, призначений для проектування, аналізу й оптимізації електромагнітних компонентів і систем.

Вирішальні програми для всього спектра електромагнітних полів доступні в єдиному користувацькому інтерфейсі пакета CST Studio Suite. Ці вирішальні програми можна поєднувати для гібридної симуляції, що дає інженерам можливість ефективно й швидко аналізувати цілі системи, що полягають із безлічі компонентів. Спільне проектування з використанням інших продуктів SIMULIA дозволяє інтегрувати ЕМ-симуляцію в процес проектування й управляти процесом розробки із самих ранніх етапів.

До найпоширеніших предметів ЕМ-аналізу ставляться продуктивність і ефективність антен і фільтрів, електромагнітна сумісність і перешкоди (EMC/ЕМП), вплив людського тіла на електромагнітні поля, електромеханічні ефекти у двигунах і генераторах, а також теплові ефекти в пристроях високої потужності.

CST Studio Suite використовується в провідних технологічних і інженерних компаніях по усьому світу. Цей застосунок забезпечує значні переваги на ринку, скорочуючи цикли розробки й витрати. Симуляція дозволяє використовувати віртуальне прототипування. Можна оптимізувати продуктивність пристрою, виявляти потенційні невідповідності нормативним вимогам і усувати їх на ранніх етапах процесу проектування, зменшити кількість необхідних фізичних прототипів і звести до мінімуму ризик помилок і відкликань продукції

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 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 трохи 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 мів програмування й цільових платформ, мів 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 оцінювання якості каналів зв'язку MIMO для 5G.

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

Продовження тенденції збільшення попиту на послуги мобільної високошвидкісної передачі даних, з одного боку, і рівень розвитку технологій мобільному зв'язку – з іншої, визначають основні вимоги до мереж мобільного зв'язку покоління 5G і формують їхній вигляд. Характерною рисою вимог до мереж мобільного зв'язку 5G є реалізація наступних цільових показників ефективності [1, 2]:

– досягнення пікової швидкості передачі даних до 20 Гбіт/с на лінії вниз – від базової станції (БС) до абонентської станції (АС) – DL, і 10 Гбіт/с на лінії нагору від АС до БС – UL;

– забезпечення швидкості передачі даних 100 Мбіт/с одночасно для багатьох користувачів в умовах мегаполісів;

– підвищення спектральної ефективності до 30 біт/с/Гц для DL і 15 біт/с/Гц для UL;

– одночасне підключення декількох сотень тисяч бездротових датчиків;

– досягнення надвисокої надійності мережі (у деяких випадках імовірність успішної доставки пакетів протягом 1 мс повинна досягати 99,9999%); – зниження часу затримок на рівні керування в мобільному зв'язку до 5 мс; у супутниковому зв'язку – до 600 мс для випадку поширення в умовах високої навколоземної орбіти, 180 мс для середньої навколоземної орбіти й 50 мс для низької навколоземної орбіти;

– збереження QoS при швидкості пересування мобільних терміналів до 500 км/год;

– збільшення ємності мережі до 1 000 000 терміналів на 1 км^2.

Одне з головних вимог для реалізації нових послуг у мережах мобільного зв'язку покоління 5G – істотне збільшення пропускної здатності в порівнянні з мережами попередніх поколінь. Якщо в поколіннях 3G і 4G це досягалося в основному за рахунок впровадження нових сигнально-кодових конструкцій, оптимального розподілу частотного ресурсу (ЧР) і розширення частотного спектра, то в цей час спостерігається гострий дефіцит ЧР, а можливості його розширення досить обмежені. Розвиток технологій мобільному зв'язку в таких умовах змушує шукати нові шляхи й вирішувати складні технічні завдання, що забезпечують виконання вимог по збільшенню пропускної здатності мереж 5G. Одним з ефективних способів підвищення пропускної здатності є технологія просторового мультиплексування MIMO (multiple input multiple output). У поколіннях 3G і 4G технологія MIMO побрала свій початок із застосуванням антенних систем у конфігураціях 2x2 і 4x4 і довела свою ефективність. Однак цих конфігурацій уже недостатньо для реалізації представлених вище вимог, у зв'язку із чим організації 3GPP [3] і IEEE [4] ухвалили рішення щодо розробки наступного покоління багатоантенних систем – Massive MIMO (в англомовній літературі також зустрічаються терміни large-scale antenna system, very large MIMO, hyper MIMO). Ці системи мають потенціал для значного поліпшення таких показників, як надійність каналу зв'язки, спектральна й енергетична ефективність [5, 6, 7, 8]. Практична оцінка показників якості систем Massive MIMO була проведена в роботі [9], результати якої підтвердили реалізуємість теоретичних переваг технології в умовах реальних каналів. Одержання очікуваного ефекту від застосування Massive MIMO при розгортанні мереж 5G неможливо без інтелектуальної підтримки з боку сучасних спеціалізованих програмних засобів планування й оптимізації мереж зв'язки. Одним з таких програмних засобів є ONEPLAN/ONEGA RPLS [10].

У цій роботі розглядаються як переваги, так і складності реалізації систем Massive MIMO, а також їх особливості.

Переваги систем Massive MIMO

Виграш у просторовому рознесенні. Головна перевага Massive MIMO перед традиційними системами MIMO полягає в можливості адаптивного діаграмоутворення або формування безлічі променів (beamforming) [11] у багатокористувацькому режимі (multi-user Massive MIMO), при якому потоки даних передаються заздалегідь призначеним користувачам [12]. При цьому однієї АС виділяються ті ж самі ресурсні блоки, що й іншим АС у межах однієї стільник. За допомогою такого рознесення в просторі досягається економія часового-частотно-тимчасового ресурсу або, іншими словами, підвищення спектральної ефективності. Крім цього, так як потенційна точність концентрації спрямованого променя на заданий мобільний термінал досить висока, інтерференційні перешкоди між променями повинні значно зменшуватися. За рахунок зниження інтерференційних перешкод більш не потрібні (або потрібні в досить малих обсягах) складні механізми ортогоналізації сигналів, що значно спрощує попередню обробку сигналів і підвищує енергетичну ефективність [13]. Ключові переваги системи Massive MIMO – енергетична й спектральна ефективність.

Енергетична ефективність. Аналіз робіт [5-8, 11, 14] показує, що системи Massive MIMO у порівнянні із традиційними системами MIMO значно виграють в енергетичній ефективності. Енергетична ефективність характеризується кількістю біт у секунду, що доводяться на 1 Вт потужності сигналу, віднесеної до спектральної щільності шуму [15]. Отже, чим вище енергетична ефективність, тем менше відношення сигнал/шум потрібно для передачі одного біта даних. Застосування в системах мобільного зв'язку технологій Massive MIMO, що володіють високою енергетичною ефективністю, дозволяє знизити енергоспоживання устаткування, поліпшити електромагнітну сумісність за рахунок зниження випромінюваної потужності, а також підвищити екологічну безпеку передавальних радіотехнічних об'єктів, особливо АС [16].

Доведене, що при збільшенні на БС кількості антен у системі Massive MIMO до величини M, потужність передачі кожної АС можна зменшити пропорційно 1/M [8]. Дане твердження слушне при ідеальних вимірах стану каналу на БС, тобто якщо індикатори, що втримуються в групі інформації CSI (channel state information), повною мірою характеризують стан радіоканалу й відомий на БС. Якщо ж індикатори CSI не ідеальні, потужність передачі кожної АС можна знизити пропорційно [8]. Безумовно, це менше, чим у першому випадку, однак виграш суттєвий і при цих умовах.

Спектральна ефективність характеризується швидкістю передачі даних на одиницю ширини спектра частот і виміряється відношенням швидкості передачі даних на 1 Гц використовуваної смуги частот (біт/с/Гц). Пропускна здатність системи Massive MIMO збільшується пропорційно кількості антенних портів у системі.

Згідно з моделюванням і оцінці в [5, 8], можна відзначити значні переваги систем Massive MIMO по показникові спектральної ефективності. Наприклад, для випадку використання смуги частот 20 МГц були отримані наступні результати: кожний абонентський термінал у стільнику має можливість одержати швидкість передачі даних до 17 Мбіт/с; загальна пропускна здатність стільник становить 730 Мбіт/с, що відповідає спектральній ефективності 36,5 біт/с/Гц.

Практичні виміри спектральної ефективності в [17] показали високі на сьогоднішній день результати: 79,4 біт/с/Гц (при наявності 128 антен на БС і смузі частот 20 МГц), що еквівалентно швидкості передачі даних 1,59 Гбіт/с.

Якщо зрівняти ці показники із середнім значенням для технології MIMO 4x4 в LTE-advanced [18], що становлять 3,7 біт/с/Гц, то перевага Massive MIMO по спектральній ефективності стає очевидним.

Поліпшення надійності каналу (channel harderning) є ще однією відмітною гідністю систем Massive MIMO [13], заснованим на просторовому рознесенні. Термін channel harderning (дослівно "зміцнення каналу") уже зустрічався в англомовній літературі в [19] для опису властивості поліпшення надійності

каналів зв'язки в класичних системах MIMO і в цей час використовується для систем Massive MIMO. Відомо, що просторове рознесення сприяє зниженню несприятливих впливів швидких завмирань на канал передачі. Рвсе, імовірність того, що всі рознесені в просторі канали піддаються впливу швидких завмирань для випадку незалежності й статистичної однорідності каналів, може бути описана вираженням [20]: Рвсе = P1M, де P1 – імовірність схильності одноканальної системи зв’язку швидким завмиранням, M – кількість антен у системі.

Нехай, наприклад, ризик того, що один канал в одноантенній системі може бути уражений швидкими завмираннями, рівний P1 = 0,1; тоді, увівши в систему передачі рознесення шляхом збільшення кількості антен до M = 128, знизимо ймовірність того, що всі канали в системі виявляться уражені швидкими завмираннями, до мізерно малої величини: Рвсе = 0,1128.

Складності реалізації

Крім переваг, системи Massive MIMO відрізняються й складністю технічної реалізації.

Витрати канального ресурсу на заголовки CSI. Побудова мережі радіодоступу 5G планується здійснити на наявній в LTE структурі ресурсних блоків і службових індикаторів [21], тому необхідно враховувати обмеження, які вона накладає. Так як число антен значно зростає, збільшується й кількість передавальних каналів. Отже, щоб забезпечити необхідна якість передачі й побудувати коректну матрицю прекодування (попереднього кодування) для каналів Massive MIMO, потрібно збільшити й число індикаторів PMI (precoding matrix indicator), переданих абонентським терміналом по каналу зворотного зв'язку нагору. Також буде потрібно більше ресурсу на лінії вниз (DL) для передачі сигналів CSI-RS (channel state information – reference signal), необхідних для операцій прекодування на БС. Крім цього, для коректної роботи планувальника БС при використанні режимів multi-user будуть потрібні й більші витрати радіоресурса для передачі сигналів DM-RS (demodulation reference

signal), які відправляються абонентським терміналом нагору в напрямку до БС (UL). Таким чином, значна частина радіоресурсів піде на службову інформацію. Наприклад, при використанні 64 антенних портів більш половини елементів одного ресурсного блоку піде тільки на передачу CSI-RS сигналів [13].

Погіршення якості пілот-сигналу при використанні дуплексного режиму TDD. Реалізація режиму Massive MIMO для тимчасового дуплекса має певні переваги, однак приводить до погіршення якості пілот-сигналів різних АС через негативні впливи один на одного. Ця проблема одержала назву в англійській літературі pilot contamination (дослівно "забруднення пілот-сигналу").

При використанні режиму TDD (time division duplex) у технології Massive MIMO необхідно враховувати два факти: перший – передача пілот-сигналу в TDD здійснюється по лінії нагору від АС; другий – точність виміру стану каналу (інформація про стан каналу втримується саме в пілот-сигналі) у технології Massive MIMO є ключовою умовою для адаптивного діаграмоутворенния. Ці два факти визначають складність проблематики "забруднення пілот-сигналу", яка полягає в наступному: БС одержує пілот-сигнал від АС і формує промінь убік цієї запитуючої АС, але якщо цей пілот-сигнал був уражений інший АС, то виникає небезпека інтерференції між променями вниз [13].

На першому етапі обоє абонентських термінала посилають свої пілот-сигнали, використовуючи той самий частотний ресурс. Прийнявши пілот-сигнал від АС 1, БС 1 визначає стан каналу. Одночасно до БС 1 приходить пілот-сигнал від АС 2, що приводить до виникнення перешкод і погіршенню якості пілот-сигналу АС 1. Так як БС 1 одержала два пілот-сигналу, то направляє два промені одночасно на дві АС: перший промінь – на призначену для нього АС 1 і другий промінь – помилково на АС 2, що перебуває в сусідньому стільнику. Але так як АС 2 у цей момент обслуговується власної БС 2, яка також направляє "свій" промінь на АС 2, той помилковий напрямок променя із БС 1 на непризначений для цього термінал АС 2 служить причиною перешкод і погіршення параметрів

пілота-сигналу. Очевидно, що здійснення Massive MIMO у режимі TDD вимагає розробки ефективних методів боротьби із цим негативним явищем.

Особливості реалізації режимів дуплекса в Massive MIMO

Оптимальна схема дуплекса. При реалізації Massive MIMO виникає питання: яка зі схем дуплекса – TDD або FDD (frequency division duplex) – є найбільш підходящою? Для порівняння двох режимів пропонується проаналізувати наступні показники: точність діаграмоутворенния при напрямку променя до АС; витрати канального ресурсу на заголовок пілот-сигналу; підтримка високих швидкостей пересування АС [13].

Так як в схемі TDD передача вниз і нагору здійснюється на одній частоті (але в різні моменти часу), той фізичний стан каналу для UL і DL однаково в обох напрямках. Тобто не важливо, у якому напрямку зв'язку виміряються індикатори стану каналу CSI – значення будуть із великою ймовірністю рівні й для каналу нагору, і для каналу вниз. На відміну від TDD, у режимі FDD така рівнозначність неможлива, так як канали нагору й униз передаються на різних частотах, і, отже, виміру параметрів CSI для каналу вниз повинні відбуватися тільки в самому каналі, а результати цих вимірів для наступної обробки на БС можуть передаватися тільки по каналу нагору. У підсумку при такому режимі обміну параметрів CSI виникне проблема неактуальності прийнятих по такому зворотному зв'язкові даних на БС, які негативно впливають на точність діаграмоутворенния, тобто на найважливіший атрибут багатокористувацького (MU, Multi User) Massive MIMO. Виходячи із цього порівняння, можна зробити вивід, що більш підходящим методом дуплекса для Massive MIMO є TDD.

Наступний критерій порівняння – витрати канального ресурсу на пілот-сигнал. Так як в режимі FDD стан каналів униз виміряється на АС, те кожний антенний порт БС повинен передавати свій пілот-сигнал униз, тобто передач пілот-сигналу вниз буде стільки, скільки антенних портів є на БС. У режимі TDD, навпаки, стан каналу виміряється на БС, ґрунтуючись на інформації пілот-сигналу, посланого від антенного порту АС. Але АС мають, як правило, тільки

один антенний порт. Отже, той самий пілот-сигнал буде прийнятий усіма антенними портами на БС, що означає: БС зможе виміряти стани всіх каналів від однієї антени АС до багатьом антенам БС, використовуючи тільки одну передачу пілот-сигналу нагору. Із чого випливає, що кількість передач пілот-сигналу в режимі FDD відповідає кількості антенних портів на БС (порядку сотень для Massive MIMO), у той час як кількість передач пілот-сигналу в режимі TDD відповідає кількості антенних портів на АС (як правило, один антенний порт). Тому за критерієм витрат канального ресурсу на передачу пілот-сигналу більш кращим виявляється режим TDD.

Переходячи до третього критерію – надання зв'язку із заданим QoS на високих швидкостях пересування АС, до 500 км/год, необхідно сказати, що час передачі пілот-сигналу при цьому повинне бути настільки мало, щоб передана інформація про стан каналу завжди була актуальна. Але, як показано вище, і без того високі додаткові витрати канального ресурсу на пілот-сигнал у режимі FDD збільшаться ще більше при швидкому пересуванні АС внаслідок необхідності постійної актуалізації стану каналу. Напроти, режим TDD не вимагає настільки високих витрат канального ресурсу.

Підводячи підсумок про три розглянутих вище критеріях, можна зробити вивід, що більш підходящим режимом дуплекса для технології Massive MIMO є TDD.

Варіанти конструкцій антенних ґрат в Massive MIMO

Як правило, конструктив систем Massive MIMO заснований на багатоелементних антенних ґратах, побудованих по лінійній, циліндричній, прямокутній або ж розподіленій топології [3]. Результати вимірів показали [9], що, використовуючи циліндричну конструкцію, можна досягтися розпізнавання вхідних сигналів у двох вимірах, однак, застосовуючи лінійну конструкцію з таким же кількістю антенних елементів, можна добитися більшої точності азимутального дозволу, але лише в одному вимірі. В обох дослідах був обраний

діапазон 2,6 ГГц, і число антенних портів, розташованих друг від друга на відстані половини довжини хвилі, становило 128.

Правові складності реалізації систем Massive MIMO

Особливості функціонування адаптивного діаграмоутворенния можуть створити дилему при подачі заявок на її використання в наглядові органі, що контролюють як ЕМС радіоелектронних засобів, так і екологічну безпеку по електромагнітному факторі. З одного боку, якщо подавати заявку за максимальним значенням ефективно випромінюваної потужності одного вузького променя, те, по-перше, можлива відмова в дозволі через перевищення норм ЕМС і екологічної безпеки, а, по-друге, картина буде недостовірної, так як це максимальне значення буде досягатися лише в певній точці стільник, при цьому в більшій частині стільник значення буде набагато нижче. З іншого боку, якщо подавати заявку з усередненим значенням, при перевірці може бути виявлене порушення, так як ефективно випромінювана потужність, реально використана в промені, буде більше заявленою. Тому така неоднозначність зажадає в майбутньому введення нових нормативних актів для одержання спеціального дозволу на використання технології Massive MIMO.

Представлені в роботі результати аналізу показали, що застосування технології Massive MIMO відкриває нові можливості й вносить істотний вклад у досягнення заявлених вимог для мереж мобільного зв'язку 5G. До очевидних переваг Massive MIMO слід віднести поліпшення надійності каналу передачі, підвищення спектральної й енергетичної ефективності й, як наслідок, зниження схильності завмиранням і виниклим із цієї причини помилкам; підвищення швидкості передачі даних і збільшення ємності мережі до значень, необхідних для реалізації мереж п'ятого покоління.

3.2 Розробка структурної схеми

Пристрій-орієнтована архітектура

Архітектура, орієнтована на базові станції (БС) стільникових систем, може змінитися в 5G. Можливо, прийшов час переглянути концепції висхідної й спадної ліній зв'язку, а також контрольних каналів і каналів даних, для поліпшення шляхів проходження інформації з різним пріоритетом і цілями убік (використання) різних вузлових пристроїв (наборів вузлів) усередині мережі.

Міліметрові діапазони

Хоча спектри відчувають дефіцит на надвисокі (мікрохвильових) частотах, є ще досить багато вільного місця в областях міліметрових хвиль. Це спектральне «Ельдорадо» привело до «золотої лихоманки»: зараз дослідники з різних областей активно вивчають різні аспекти передачі міліметрових хвиль. Будучи далеко не повністю вивченими, такі технології вже стандартизовані для сервісів малої дальності (стандарт IEEE 802.11 AD), а також розгорнуті для нішевих додатків, таких як транспортні мережі зв'язку за допомогою «малого стільника».

Масиви MIMO

Масиви MIMO (Multiple-Input/Multiple-Output) припускають використання великої кількості антен для мультиплікування повідомлень на кілька пристроїв у кожному частотно-тимчасовому джерелі, коли випромінювана енергія фокусується убік передбачуваного напрямку при мінімізації інтерференції. Масиви MIMO можуть зажадати серйозних змін в архітектурі, особливо в дизайні макробазових станцій (Macro Base Station, MBS), а також можуть привести до появи нових типів розгортання.

Інтелектуальні пристрої

Стільникові мережі 2G/3G/4G будувалися виходячи з передбачуваного повного контролю над інфраструктурою. затверджуємо, що проектувальникам систем 5G варто відкинути цей постулат і використовувати інтелект пристроїв у рамках різних рівнів стекового протоколу, наприклад дозволяючи пристроям

підключатися по інтерфейсу D2D (Device-to-Device) або використовувати мобільне смарт-кешування. Хоча ця філософія дизайну вимагає, в основному, змін на рівні вузлів (компонентні зміни), це також позначиться й на архітектурному рівні.

Вбудована підтримка M2M-взаємодії

Вбудоване включення M2M-взаємодії в 5G припускає задоволення трьох принципове різних вимог, пов'язаних з різними класами низькошвидкістних сервісів передачі даних: підтримку великої кількості низькошвидкістних пристроїв, забезпечення мінімальної швидкості передачі даних практично в будь-яких умовах і дуже малу затримку передачі даних. Задоволення цих вимог в 5G вимагає нових методів і ідей як на компонентному, так і на архітектурному рівнях.

Пристрій-орієнтована архітектура

Стільниковий дизайн історично ґрунтувався на аксіоматичній ролі «стільника» у якості основних одиниць мережі радіодоступу. При такому конструкційному постулаті сервіс пристрою надається шляхом установлення спадної й висхідної ліній зв'язку (Downlink/Uplink, DL/UL), що несуть контрольний трафік і трафік даних, через БС, що повідомляє стільнику, де саме розташований пристрій. Останні кілька років різні тенденції вказують на порушення цієї «стільниково-орієнтованої» структури:

– Щільність розміщення БС швидко росте, що обумовлене ростом гетерогенних мереж. Хоча вже при їхній наявності була стандартизовано 4G, але ця архітектура споконвічно не була призначена для такої підтримки. Ущільнення мереж може зажадати деяких серйозних змін в 5G. Розгортання БС із передавальними потужностями, що значно різняться, і зонами покриття викликає, наприклад, необхідність поділу спадної й висхідної ліній зв'язку таким чином, щоб дозволити відповідній інформації проходити через різні вузлові станції [5].

– Потреба в додатковому спектрі неминуче приводить до співіснування частотних діапазонів з кардинально різними характеристиками поширення в

рамках однієї й тієї ж системи. У цьому контексті пропонується концепція фантомних гнізд (стільника), коли рівні даних і контролю розділяються: контрольна інформація відправляється на високопотужні вузли на мікрохвильовій частоті, а корисні дані передаються на малопотужні вузли на частотах міліметрових хвиль [6].

– Нова концепція, називана «централізованої базовою смугою» (centralized baseband), пов'язана з поняттям хмарних мереж радіодоступу й застосовна там, де віртуалізація приводить до поділу між вузлами й устаткуванням, що управляють обробкою цих вузлів [7]. Апаратне забезпечення, наприклад, може динамічно виділятися в пулі різних вузлів залежно від показників, певних оператором.

– Нові класи сервісів можуть зажадати повного перегляду архітектури. Поточні роботи, якщо глянути на дизайн архітектури, ранжуються від централізації або часткової централізації (наприклад, за допомогою агрегаторів) до повного розподілу (наприклад, за допомогою стислого зондування й/або багатоскачковим образом (multihop – «передача з переприйманням»)).

– Парадигми спільних комунікацій, таких як спільні багатоточкова (Cooperative Multipoint, Comp) або ретрансляційна, які, хоча й зазнали невдачі, незважаючи на первісний ажіотаж, проте досить вигідні й можуть зажадати перегляду функцій різних вузлів [8]. У контексті ретрансляції, приміром, останні розробки в бездротовому мережному кодуванні припускають такі принципи передачі, які дають можливість відновлення деяких втрат, пов'язаних з напівдуплексною ретрансляцією [9]. Крім того, недавні дослідження вказують на ймовірність використання полнодуплексних вузлових точок для комунікації малої дальності вже в не настільки віддаленому майбутньому.

– Використання інтелектуальних пристроїв може вплинути й на мережі радіодоступа. Зокрема, і інтерфейс D2D, і «розумне» кешування (смарт-кешування) викликають переосмислення всієї архітектури, коли центр

притягання переміщається від ядра мережі до її периферії (пристрою, локальні бездротові проксі, ретранслятори).

Базуючись на цих тенденціях, вважаємо, що « стільниково-орієнтована» архітектура (архітектура, орієнтована на стільник) повинна трансформуватися в пристрій-центричну: будь-які, що носяться (людиною або машиною) пристрою повинні мати можливість комунікувати, обмінюючись при цьому різноманітними інформаційними потоками за допомогою декількох можливих пристроїв (приймачів) у гетерогенних вузлових точках. Іншими словами, сукупність мережних вузлів, що забезпечують можливість підключення до даного пристрою, а також функції цих вузлових точок при певному сеансі зв'язку повинні бути адаптовані до конкретного пристрою й сесії. Згідно із цією концепцією, поняття висхідної й спадної ліній зв'язку, а також канали керування й даних повинні бути переглянуті.

Хоча необхідність проривних змін у дизайні архітектури й представляється очевидної, як і раніше необхідні серйозні наукові зусилля, щоб перетворити отримані знання в послідовні й реалістичні пропозиції. Оскільки історія інновацій показує, що зміни в архітектурі часто приводять до великих технологічних розривів [1], вважаємо, що перераховані вище тенденції можуть вплинути на розвиток 5G.

З'єднання на міліметрових хвилях

Мікрохвильова стільникова система має дуже невеликий обсяг спектра: у цей час використовується діапазон шириною близько 600 МГц, і вони діляться між усіма операторами [10]. Існує два способи одержати доступ до більш широкого мікрохвильового спектра:

– Переформувати спектр. Це вже відбулося в усьому світі з перепрофілюванням ефірного телевізійного спектра убік таких додатків, як широкосмуговий доступ у сільських місцевостях. На жаль, таке перепрофілювання не звільнило досить багато спектра (усього близько 80 МГц), крім того, сполучене з високими витратами.

– Поділитися використовуваним спектром, наприклад за допомогою техніки когнітивного радіо. На когнітивне радіо споконвічно покладали більші надії, однак тепер вони відносно згасли внаслідок того, що співробітники виявилися не цілком готові до такої кооперації, що є основною перешкодою для підвищення ефективності використання спектра для вторинних користувачів.

У цілому, представляється, що кращий варіант подвоєння поточної пропускної здатності стільника зв'язаний саме з надвисокими частотами. Також існує великий спектр міліметрових хвиль у діапазоні частот 3-300 ГГц. Багато діапазонів тут представляються перспективними, у тому числі найближче локальне багатоточковий розподіл сервісів у діапазоні 28-30 ГГц, безкоштовні ліцензії в діапазоні 60 ГГц і смуги групи E2 71-76, 81-86 і 92-95 ГГц. Приблизно, кілька десятків гігагерц можуть стати доступними для 5G, що забезпечить куди більш масштабний приріст щодо того, що доступно на даний момент. Незайвим буде сказати, що робочою необхідністю є проведення такої політики в області спектрів, яка надасть ці діапазони для стільникового мобільного зв'язку.

Поширення таких хвиль не є непереборною проблемою. Останні виміри показують схожі загальні характеристики з мікрохвильовими частотами, у тому числі щодо втрат, пов'язаних з відстанню, пройденим хвилею, а також щодо можливості комунікації із пристроями поза полем видимості. Головна різниця між мікрохвилями й міліметровими хвилями полягає в чутливості до перешкод: в [11], наприклад, рівень втрат оцінюється у два рази при поширенні в поле видимості й у чотири (плюс додаткові втрати потужності) – при поширенні поза полем видимості. При дослідженнях стільникових міліметрових хвиль необхідно враховувати цю чутливість до перешкод, аналізувати більш складні моделі каналів, а також вивчити вплив таких складових, як інфраструктура високої щільності й наявність ретрансляторів. Ще одним фактором є поділ між рівнем контролю й рівнем даних, про що вже згадувалося вище.

Антенні ґрати є ключовим елементом у системах міліметрових хвиль. Більші масиви можуть бути використані для підтримки постійною антеною

апертури, усуваючи частотну залежність шляхових втрат щодо всеспрямованих антен (при використанні їх на одній стороні лінії) і забезпечуючи виграш у протистоянні великим тепловим перешкодам (шумам) для пропускної здатності мережних ґрат (при використанні по обидва боки лінії). Адаптивні ґрати з вузькими пучками променів також зменшують вплив інтерференції, а це означає, що системи на міліметрових хвилях частіше можуть працювати в умовах обмежених шумів, ніж в умовах обмежених перешкод. Оскільки повноцінне з'єднання може відбуватися тільки при достатній кількості ґрат, для цього необхідні нові протоколи випадкового доступу, коли передавачі можуть випромінювати тільки в певних напрямках, а приймачі можуть одержувати інформацію тільки з певних напрямків. Будуть потрібні також адаптивні алгоритми обробки масивів, які зможуть швидко пристосовуватися, якщо пучки блокуються перешкодами або кілька антенних пристроїв закриваються власним тілом користувача.

Системи міліметрових хвиль також мають різні апаратні обмеження. Одним з основних є високе енергоспоживання різних сигнальних компонентів, і тут мова, в основному, іде про аналого-цифрові перетворювачі (АЦП) і цифро-аналогових перетворювачах (ЦАП). Наприклад, звичайна мікрохвильова архітектура, коли кожна антена підключена до високошвидкісного АЦП/ЦАП, чи навряд може бути застосовна для міліметрових хвиль, якщо не буде зроблений величезний стрибок уперед в області напівпровідникових технологій. Однієї з альтернатив є гібридна архітектура, при якій формування променя здійснюється в аналогових радіочастотах і численні потоки променів підключаються до невеликої кількості АЦП або ЦАП; при такій альтернативі для керування аналоговими потужностями формування променів будуть потрібні відповідні алгоритми обробки сигналів. Іншою альтернативою є підключення кожної радіочастоти до однобітного АЦП/ЦАП з дуже низькою споживаною потужністю; у цьому випадку промені будуть формуватися в цифровому виді, але з досить «зашумленними» даними. Існує безліч досліджень проблем оптимізації

різних трансіверних стратегій, а також аналізу їх потенціалу, включаючи багатокористувацькі можливості й використання такої характеристики каналу, як розрідженість.

Наведене порівняння швидкостей стільникової передачі даних між мікрохвильовими системами, що використовують смугу пропущення 50 МГц (однокористувацька одинична антена й однокористувацький MIMO), і системами міліметрових хвиль зі смугою пропущення 500 МГц (одиничний користувач). Результати представлені з погляду приросту (в %) із прив'язкою до базового рівня MIMO 4×4. Більш докладно про порівняння розказане в [12]. У пояснення слід помітити, що результати, наведені в даному розділі, були отримані з урахуванням сучасного розуміння розглянутих технологій. Однак підкреслюємо, що на даний момент поки неможливо дати повністю реалістичну оцінку й провести порівняння з розгорнутими 4 G-системами. Безперечно, деякі дослідження усе ще перебувають у так званій «ажіотажній» фазі, і має бути проробити більшу роботу, перш ніж стійке розуміння функціонування 5G і необхідних допоміжних засобів зможе бути досягнуте.

Використання міліметрових хвиль забезпечує дуже високі показники в порівнянні із двома різними мікрохвильовими системами. Вигоди десятикратно перевищують ріст спектра через підвищену потужність сигналу й зниження взаємних перешкод завдяки спрямованим променям – як у приймачі, так і в передавачі.

З наведеного вище обговорення, а також посилаючись на модель Хендерсона-Кларка, містимо, що використання міліметрових хвиль зажадає радикальних змін у системі, оскільки воно вплине як на дизайн компонентів, так і на дизайн архітектури. Отже, розглядаємо використання міліметрових хвиль як потенційно проривних технологій для 5G, які, якщо вищеописані проблеми будуть вирішені, можуть привести до неперевершеної швидкості передачі даних і зовсім іншому досвіду користувачів.

Масиви MIMO

Масиви (також називані «великомасштабні MIMO» або «великомасштабні антенні системи») є однієї з форм систем багатокористувацького MIMO, у якому число антен на БС набагато більше, ніж кількість пристроїв для джерел сигналів [14]. Сприятливий вплив закону більших чисел згладжує частотні залежності в каналі й, у цілому, дає величезний виграш у спектральній ефективності, яка тільки може бути досягнута. Наведене порівняння швидкостей стільникової передачі даних для додатків фіксованого доступу масивної MIMO. Масиви з 2048, 4096 або 8192 антен, що використовують 50 МГц і випромінюючих у цілому 120 Вт, обслуговують 1000 користувачів, розташованих випадковим образом у секції радіусом 6 км. Результати наведені в термінах (з погляду) приросту (в %) із прив'язкою до базового рівня MIMO 4×4.

У контексті рамок моделі Хендерсона-Кларка затверджуємо, що масивні MIMO мають проривної потенціал для 5G по наступних причинах:

– На рівні вузлів це масштабована технологія, яка тем і відрізняється від 4G, у якій подальша секторизація нездійсненна, оскільки існує обмежений простір для великогабаритних азимутально-орієнтованих антен і є неминучий кут поширення. У свою чергу, однокористувацька MIMO має обмеження на кількість антен, які можуть уміститися в певних мобільних пристроях. При цьому практично не існує обмежень на кількість антен БС у масивних MIMO – за умови, що розділене в часі дуплексування використовується для того, щоб дозволити розрахувати канал за допомогою керування висхідною лінією зв'язку.

– Технологія 5G дозволить впровадити певні нововведення й нову архітектуру. Хоча й можливо уявити собі пряму заміну макро-БС із ґратами резонансних антен низького коефіцієнта підсилення, слід розуміти, що можливо також розгортання, наприклад, рівнокутних ґрат на фасадах хмарочосів або на поверхні резервуарів для води в сільських місцевостях. Крім того, ті ж принципи масивів MIMO, що регулюють використання розставлених антенних ґрат, застосовуються також до розподілених дислокацій, коли, наприклад, кампус

коледжу або ціле місто може бути покритий безліччю розподілених антен, які в сукупності служать багатьом користувачам (у цій конструкції централізована полосна концепція, представлена раніше, є важливим архітектурним механізмом реалізації).

Будучи дуже перспективними, масиви MIMO, однак, представляють і ряд проблем для досліджень. Розрахунки каналу є критичним, і в цей час він служить основним джерелом обмежень. Пересування користувача встановлює кінцевий когерентний інтервал, протягом якого інформація про канал повинна бути отримана й використана, і, отже, існує кінцеве число ортогональних стартових послідовностей, які можуть бути виділені пристроям. Повторне використання стартових послідовностей викликає попередні забруднення й перешкоди зв'язки, які ростуть разом з кількістю антен так само швидко, як і корисні сигнали. Зм'якшення наслідків такого забруднення є темою активних досліджень. Крім того, існує ще багато тонкощів для вивчення поширення масивів MIMO, хоча поки що досвіди підтверджують гіпотезу квазі-ортогональности каналу.

З погляду застосування, масиви MIMO можуть потенційно реалізовуватися за допомогою модульного недорогого малопотужного устаткування, коли кожна антена функціонувала б напівавтономно, але як і раніше потрібні значні зусилля, щоб продемонструвати економічну ефективність даного застосунку. Відзначимо, що на надвисоких частотах, розглянутих у цьому розділі, вартість і споживання енергії АЦП/ЦАП помітно нижче, чим на частотах міліметрових хвиль.

З обліком вищевикладеного, дійшли висновку, що прийняття масивних MIMO для 5G може являти собою великий крок стосовно до сьогоднішнього стану системи й конструкції компонентів. Щоб виправдати ці серйозні зміни, прихильникам масивів MIMO необхідно продовжити роботу з застосунку завдань, викладених вище, щоб показати реальні поліпшення продуктивності шляхом теоретичних досліджень, моделювання й стендових експериментів.

Стільникові оператори в усьому світі активно придивляються до мереж п'ятого покоління. Комерційна експлуатація очікується не раніше 2021 – 2023 років. Компанія Huawei разом з японським оператором зв'язку NTT docomo уже випробували в польових умовах зв'язок 5G і на частотах нижче 6 ГГц їм удалося добитися швидкості передачі даних в 3,6 Гбіт/с. Згідно з офіційними документами 5G максимально можлива швидкість становить до 20 Гбіт/с.

МТС і Ericsson почнуть рух у цьому напрямку в 2021 році в частотному діапазоні 5 ГГц, адже для його використання дозволів не потрібно, якщо робота ведеться усередині приміщень. Одночасно буде проведене тестування технології Ericsson Lean Carrier, що дозволяє ефективно розподіляти сигнальний трафік між стільником. Також буде проведена демонстрація нових інтерфейсів для підключення різних пристроїв до інтернету (M2M): EC-GSM (Extended Coverage GSM), LTE-M (LTE Machine) і NB-IoT (Narrow band Lot LTE based).

Цікаво, що технічні вимоги до 5G ще не сформовані – Міжнародний союз електзастосунку визначив їх тільки в 2019-2020 рр. Але вже очікується, що в підсумку для 5G будуть виділені діапазони понад 6 ГГц, які ще не зайняті існуючими технологіями. В 2021 р. МТС і Ericsson запланували створення тестової зони 5G у діапазоні 15 ГГц. Сторони очікують, що регулятор виділить їм частоти в даному діапазоні.

Так як частотні діапазони є стандартними й здебільшого покриваються існуючими розв'язками на ринку, то виробники тестового устаткування вже випустили спеціалізоване програмне забезпечення до своїх аналізаторів сигналів і спектра. Наприклад, компанія Rohde & Schwarz розробила застосунок, який дозволяє оцінити спектр сигналу 5G. Застосунок складається з аналізатора спектра й сигналів FSW, векторного генератора сигналів і нового програмного забезпечення R&S TS-5GCS. Користувачі можуть вибрати необхідні моделі генератора й аналізатора з лінійки приладів, беручи до уваги необхідний частотний діапазон.

Технологія MIMO буде використовуватися й у мережах 5G, але при цьому якщо в 4G використовується MIMO 2x2, то в технології 5G число антен повинне збільшитися. Ця технологія має відразу два вагомі аргументи для застосування: швидкість передачі даних зростає практично пропорційно кількості антен, при цьому якість сигналу поліпшується за рахунок приймання сигналу відразу декількома антенами. Технічні фахівці при тестуванні мереж п'ятого покоління повинні мати можливість виміру імпульсної характеристики каналу MIMO (зондування каналу), що є основним завданням при вимірі каналу MIMO. Отримані дані вимірів надалі використовують при моделюванні каналу MIMO.

Процес зондування каналу дає можливість оцінити характеристики каналу передачі або каналу мобільного радіозв'язку при встановленні передачі даних (Channel sounding). Це досить складний процес, який зажадає точного устаткування синхронізованого за часом, що працює в міліметровому частотному діапазоні.

Імпульсна характеристика каналу

***h*(*t*) – CIR для вилучення потрібна обробка сигналів після або в режимі реального часу як використання методу перехресної кореляції у цій формулі**

$$R_{xy}(\tau) = E[x^*(t)y(t-\tau)]$$
$$= h(t-\tau) \otimes R_x(\tau)$$
$$\approx h(t-\tau) \otimes \delta(\tau) = h(t)$$

Рисунок 3.1 – Структурна схема системи

Щоб виміряти параметри радіоканалу MIMO необхідно забезпечити синхронізацію передавача й приймача вимірювальної установки для зондування каналу. Для забезпечення синхронізації застосовуються наступні методи:

– синхронізація по кабельному з'єднанню – даний метод використовують при проведенні вимірів усередині приміщень;

– синхронізація по прийнятому сигналу – простий спосіб, але підданий великому впливу перешкод і викривлень у каналі, має низьку точність;

– синхронізація по сигналах GPS – приймачі GPS дозволяють використовувати стандартні сигнали PPS або 10 МГц у якості опорних сигналів для синхронізації;

– використання високостабильних опорних генераторів, наприклад, на основі рубідію. Рубидіеві генератори синхронізуються на початковому етапі перед проведенням вимірів і зберігають високу стабільність частоти протягом декількох годин.

Характеристики радіоканалу залежать від безлічі параметрів, наприклад: втрати, відбиття, поглинання, множинне поширення радіо сигналу, ефект Доплера і т.д. Вимір імпульсних характеристик каналу проводиться по кореляційному принципу. Векторний генератор сигналів використовується як джерела сигналу для зондування каналу й формує тестовий псевдовипадковий сигнал з гарними кореляційними властивостями, а аналізатор сигналів і спектра FSW виступає в ролі приймача.

3.3 Розробка функціональної схеми

Програмне забезпечення, розроблене в даній роботі – універсальне рішення для оцінювання якості каналів зв’язку MIMO для 5G. Дана система знайде застосування в більшості мобільних операторів і виробників устаткування. Програмне забезпечення, розроблене в даній роботі, призначено для планування мобільної мережі, оптимізації існуючих мереж зв'язку й керування

продуктивністю. Також комплекс дозволяє здійснювати 3D-моделювання зони покриття, проектування малих стільникових і гетерогенних мереж (HetNet).

Характеристики програмного забезпечення, розробленого в даній роботі:

– Повна підтримка мереж 5G : 5G і 5G NR.

– Просунутий симулятор для моделювання OFDM технології.

– Підтримка складних мультидіапазоних антенно-фідерних систем.

– Вбудована повнофункціональна геоінформаційна система (ГІС) MapInfo Professional.

– Планування гетерогенних мереж зв'язку.

– Підтримка Windows: Windows Vista 64 bits, Windows 7 Pro 64 bits, Windows 8 Pro 64 bits, Windows 10.

– Підтримка версій Windows-сервер: Windows Server 2003 & 2003 R2 64 bits, Windows Server 2008 64 bits, Windows Server 2008 R2 64 bits, Windows Server 2012 64 bits, Windows Server 2012 R2 64 bits.

– Підтримка Sentinel 8.5.5.

Компанії можуть використовувати програмне забезпечення, розроблене в даній роботі, на будь-якому етапі життєвого циклу мобільної мережі для стратегічного планування й проектування, керування мережею, перепланування й оптимізації, необхідної у випадку використання встаткування різних постачальників і застосування різних технологій.

Програмне забезпечення, розроблене в даній роботі, надає доступ до інформації про конфігурацію, параметри й продуктивність мережі, а також візуалізує дані геолокації викликів у режимі реального часу. Система веде облік і забезпечує технічних фахівців даними про поточний стан мобільної мережі, інформацією про зміну вимог у трафіку й виникаючих проблемах. Це дозволяє скоротити час на виявлення областей з низькою продуктивністю й полегшує процес планування й оптимізації.

Можливості комплексу дозволяють мобільним операторам підвищити ефективність проектування мобільної мережі й прискорити процес переходу до

мереж наступного покоління. У результаті програмне забезпечення, розроблене в даній роботі, скорочує Ваші витрати на зміст мережевої інфраструктури.

Ключові пеерваги програмного забезпечення, розроблене в даній роботі:

– Простота використання – швидке створення проекту, генерація розрахунків радіопокриття й симуляція більшості параметрів мережі.

– Продуктивність – висока продуктивність моделі даних, швидкий час відгуку користувальницького інтерфейсу.

– Маштабуємість – можливість роботи з величезними проектами завдяки оптимізації використання пам'яті.

– Стабільність – висока якість ПЗ, засновано на сучасній архітектурі .NET Framework.

– Облік майбутніх технологій – нова модель даних з використанням складних антенних систем (таких як shared, RET, E-azimuth, smart antennas, MIMO і ін.).

Всі інструменти логічно згруповані й перебувають у стрічці над основними робочими вікнами. Всі необхідні інструменти можуть бути розташовані на панелі швидкого доступу.

Підтримується 64-бітна версія MapInfo Pro 15.2. Особливістю даного релізу є підтримка сенсорних екранів і жестів, удосконалений конструктор звітів, підвищена продуктивність операцій з таблицями й графічними об'єктами.

Програмне забезпечення, розроблене в даній роботі, дозволяє сполучення с VistaNEO (на підставі алгоритмів OSS), 3D геолокацію трафіку, реалізована підтримка ідентифікатора IMSI в 5G і транзитних вузлів 5G. Також серед корисних функцій буде аналіз параметрів PCI (Physical Cell Identifier) і статичний режим в інструменті діалогової обробки.

Розглянемо функції програмного забезпечення, розробленого в даній роботі.

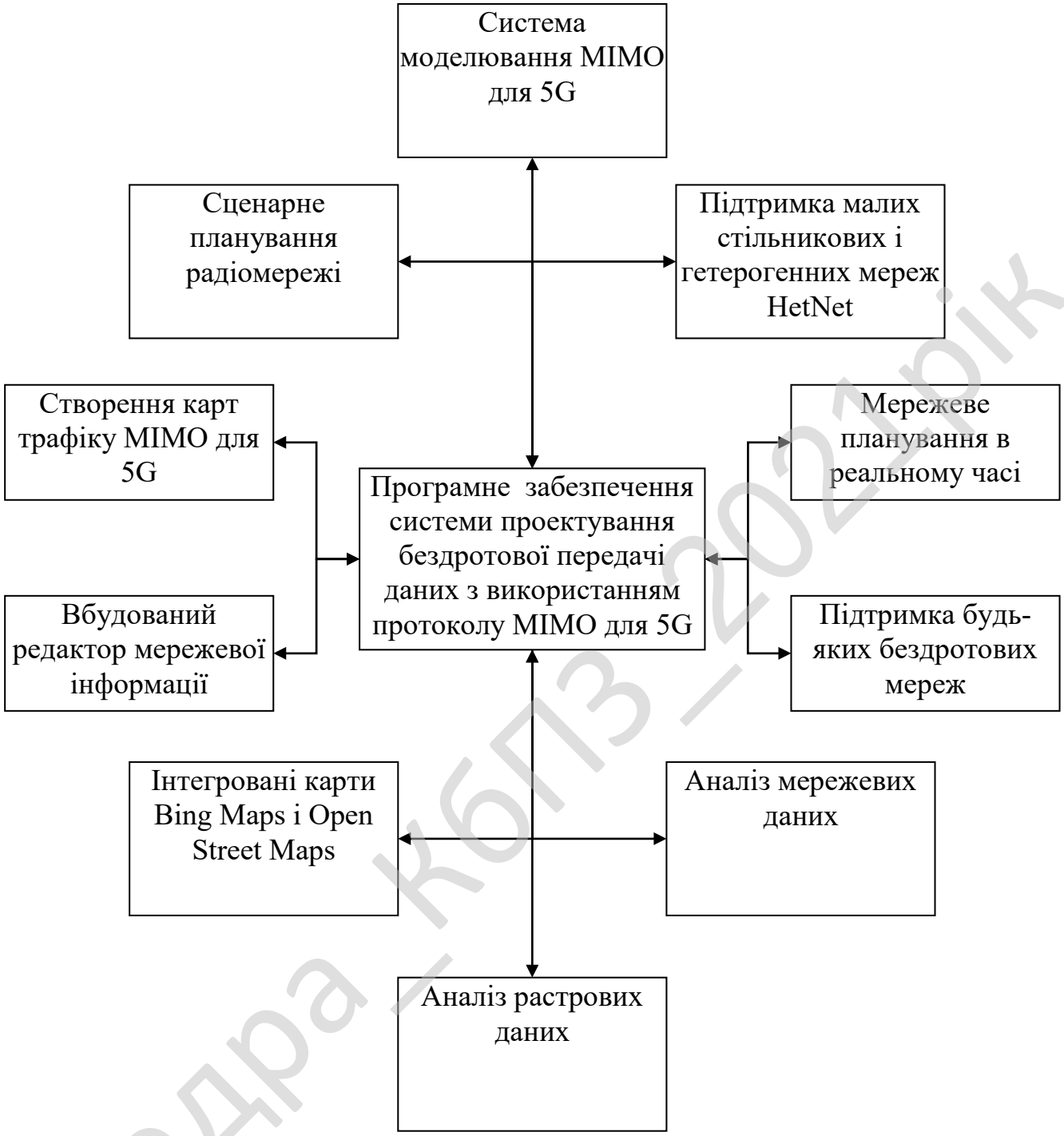

Рисунок 3.2 – Функціональна схема системи

Система моделювання MIMO для 5G

Програмне забезпечення, розроблене в даній роботі, пропонує інструмент моделювання MIMO для 5G, вибрати які можна через інтернет або знайти в інтегрованій бібліотеці:

– Моделювання всією антеною системи здійснюється в одному файлі (.pafx file).

– Підтримка мульти-діапазонних антен. eTilt, eBeamwidth і eAzimuth.

– Підтримка просунутих антенних систем. Підтримка розносу Tx/Rx і MIMO.

– Моделювання SMART антен: beam-forming і багатовібраторні антени MIMO для 5G.

– Фізичні характеристики антени MIMO для 5G.

– Тривимірне моделювання антен MIMO для 5G.

Створення карт трафіку MIMO для 5G

Карти трафіку забезпечують основу для сучасного радіопланування, показуючи місця концентрації навантаження на мережу. Найважливіші дані для оцінки роботи мережі під навантаженням. Карти трафіку можуть бути засновані на прогнозах або статистику. Можливе використання різних трафікових моделей для різних сценаріїв розвитку мережі або для різних сервісів:

– Детальне моделювання абонентів.

– Визначає порядок доступу абонентів у мережу, і моделі їхнього поводження (сервіс, пріоритет, устаткування).

Вбудований редактор мережевої інформації

Вбудований редактор мережевої інформації, оснащений функціоналом, аналогічним Excel. Забезпечує масове редагування даних, копіювання/вставку в/з Excel. Змінює параметри планованої мережі на льоту, перевіряючи їх на наявність помилок. Більше немає необхідності готовити мережеві дані для проекту в Excel.

Аналіз растрових даних

Інтегрований модуль Vertical Mapper для керування растровою інформацією:

– Стандартний і відкритий растровий формат.

– Гнучкі можливості по відображенню.

– Можливість редагування карт.

– Можливості по обробці растрової інформації (додавання й т.д.).

Інтегровані карти Bing Maps і Open Street Maps

Повністю інтегрований функціональний модуль підзавантаження інтернет-карт. Забезпечується візуалізація будь-якого розрахункового шару Програмне забезпечення, розроблене в даній роботі, на підложці з високоточної картографічної інформації про міську забудову. Є можливість редагування об'єктів прямо з Bing Maps Viewer.

Аналіз мережевих даних

У програмному забезпеченні, розробленому в даній роботі, може бути завантажена статистика по кожному секторі із зовнішніх джерел. До кожного проекту мережі можуть бути прив'язані кілька наборів даних (наприклад, ключові показники або статистика за різні тимчасові періоди). Зовнішні дані можуть бути відбиті в графічному виді, включені в статистику, або використовуватися як підстава для автоматичної фільтрації. Може бути представлений зв'язок між мережевими даними й розрахунками.

Підтримка будь-яких бездротових мереж

Програмне забезпечення, розроблене в даній роботі, підтримує більшість бездротових стандартів, розгорнутих у цей час, включаючи GSM, GPRS, EDGE, WCDMA, HSPA, HSPA +, 5G (TDD і FDD), 5G NR, Wi-Fi, WiMAX, cdma2000, EVDO, iDEN, TETRA, P25 і TDMA / FDMA системи з одночасною передачею.

Можливість мережевого планування в реальному часі для досягнення кращих результатів

Можливості планування мережі в реальному часі дозволяє інженерам оперативно одержувати інформацію про конфігурацію мережі, фактичному навантаженню й продуктивності.

Підтримка малі стільникові і гетерогенні мережі HetNet

Програмне забезпечення, розроблене в даній роботі, пропонує повний спектр функцій, необхідних для розгортання малих стільникових і гетерогенних

мереж HetNet, дозволяє робити стратегічне планування й оцінку окупності інвестицій (ROI).

Призначений для сценарного планування радіомережі

Програмне забезпечення, розроблене в даній роботі, пропонує можливість створення одночасно декількох сценаріїв розвитку мобільної мережі. Це дає можливість операторам розглянути й зрівняти різні проекти й вибрати кращий.

Висока економічна ефективність планування й розгортання 2G, 3G, 5G і 5G NR мереж

Система програмного забезпечення, розроблене в даній роботі, підтримує більшість бездротових стандартів зв'язку. Від моменту появи на світло дотепер завдяки втіленню повної підтримки 5G NR, програмне забезпечення, розроблене в даній роботі, залишається лідером в області радіопланування й оптимізації бездротових мереж, продовжуючи задовольняти потреби ринку.

Планування мобільних мереж у реальному часі

Планування й оптимізація в режимі реального часу надає інженерам можливість домогтися максимальної продуктивності й рівня якості обслуговування в мережі. Геолокація викликів, дані про продуктивність мережі й мережеві конфігурації дозволяють планувати й оптимізувати малі стільники або гетерогенні мережі HetNet. За допомогою цього можна акцентувати увагу й направляти ресурси для поліпшення в тих областях, де це буде мати найбільшу віддачу.

Правильне керування мережею для досягнення максимальної ефективності бездротової мережі

Керування існуючою бездротовою мережею – не просте завдання. Сучасні рішення по плануванню мережі є важливою частиною глобальних систем підтримки операцій/підтримки бізнесу (OSS/BSS). Програмне забезпечення, розроблене в даній роботі, має унікальні властивості, які допомагають операторам управляти своїми бездротовими мережами. Серед них відкритість системи, можливість розширення й програмуємість. Таким чином, програмне забезпечення, розроблене в даній роботі, може використовуватися як корпоративне багатокористувальницьке рішення з вичерпними можливостями.

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.

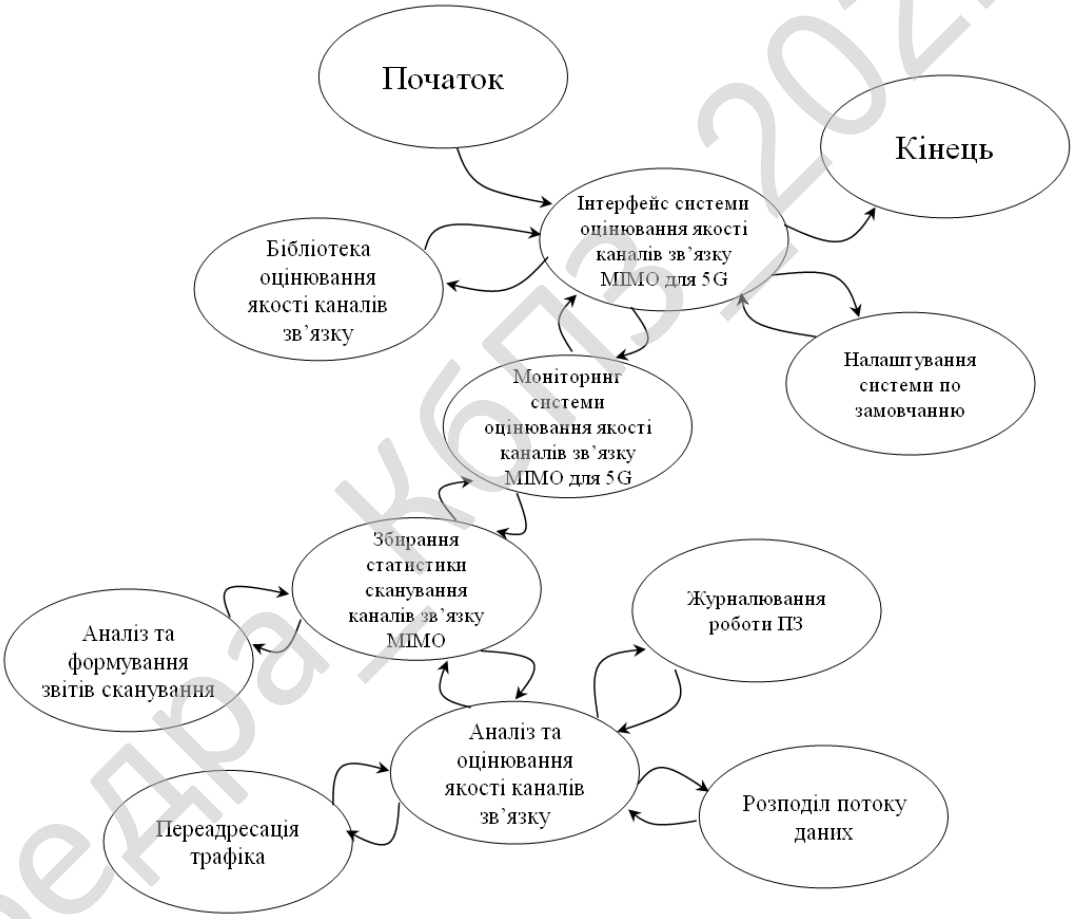

Рисунок 3.3 – Діаграма взаємодії процесів

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.

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

Під час роботи над програмою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується. Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо.

Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

Розглянемо алгоритм роботи основної програми у вигляді блок-схеми зображеної на рисунку 4.1. Вона складається з наступних функціональних блоків:

– Виведення вікна системи оцінювання якості каналів зв’язку MIMO.

– Запит зміни критеріїв оцінки якості.

– Введення даних та початкового статусу каналів зв’язку.

– Підпрограма зміни критеріїв оцінки якості каналів зв’язку (рис. 4.2).

– Виведення встановлених змін.

– Запит статистики роботи.

– Моніторинг каналів зв’язку MIMO.

– Збереження результатів у таблицю статистики БД.

– Виведення отриманих результатів.

– Запит оцінки якості каналів.

– Підпрограма оцінки якості каналів зв’язку.

– Виведення результатів аналізу якості.

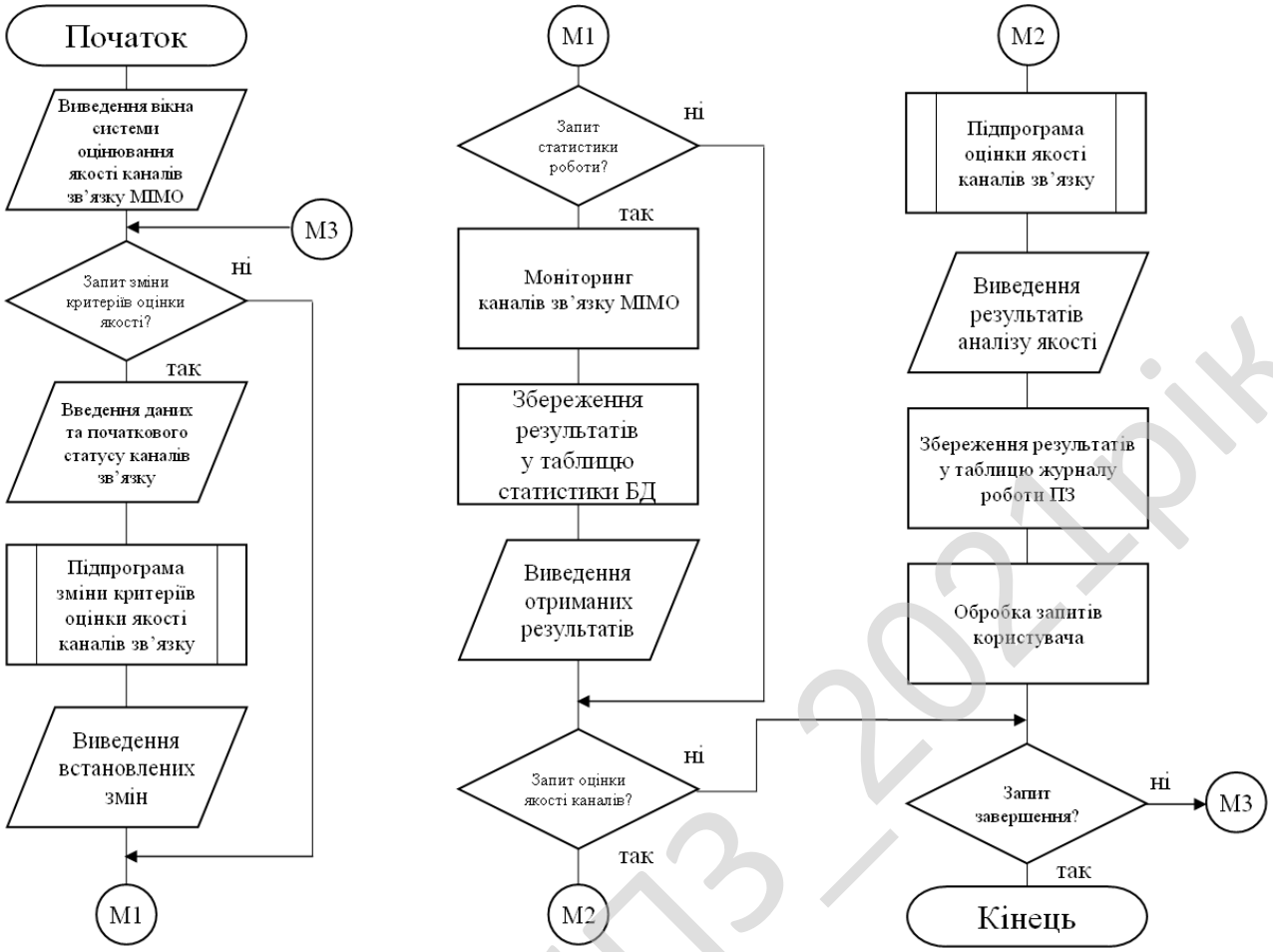

Рисунок 4.1 – Блок-схема основної програми

– Збереження результатів у таблицю журналу роботи ПЗ.

– Обробка запитів користувача.

– Запит завершення.

На рисунку 4.2 зображено роботу підпрограми з наступними функціональними блоками:

– Ініціалізація ресурсів модуля системи оцінювання якості каналів зв'язку MIMO для 5G.

– Запит оцінювання якості.

– Очищення та оновлення записів таблиці якості каналів зв'язку.

– Формування тестових запитів у канали зв'язку.

– Відправлення запитів у канали зв'язку.

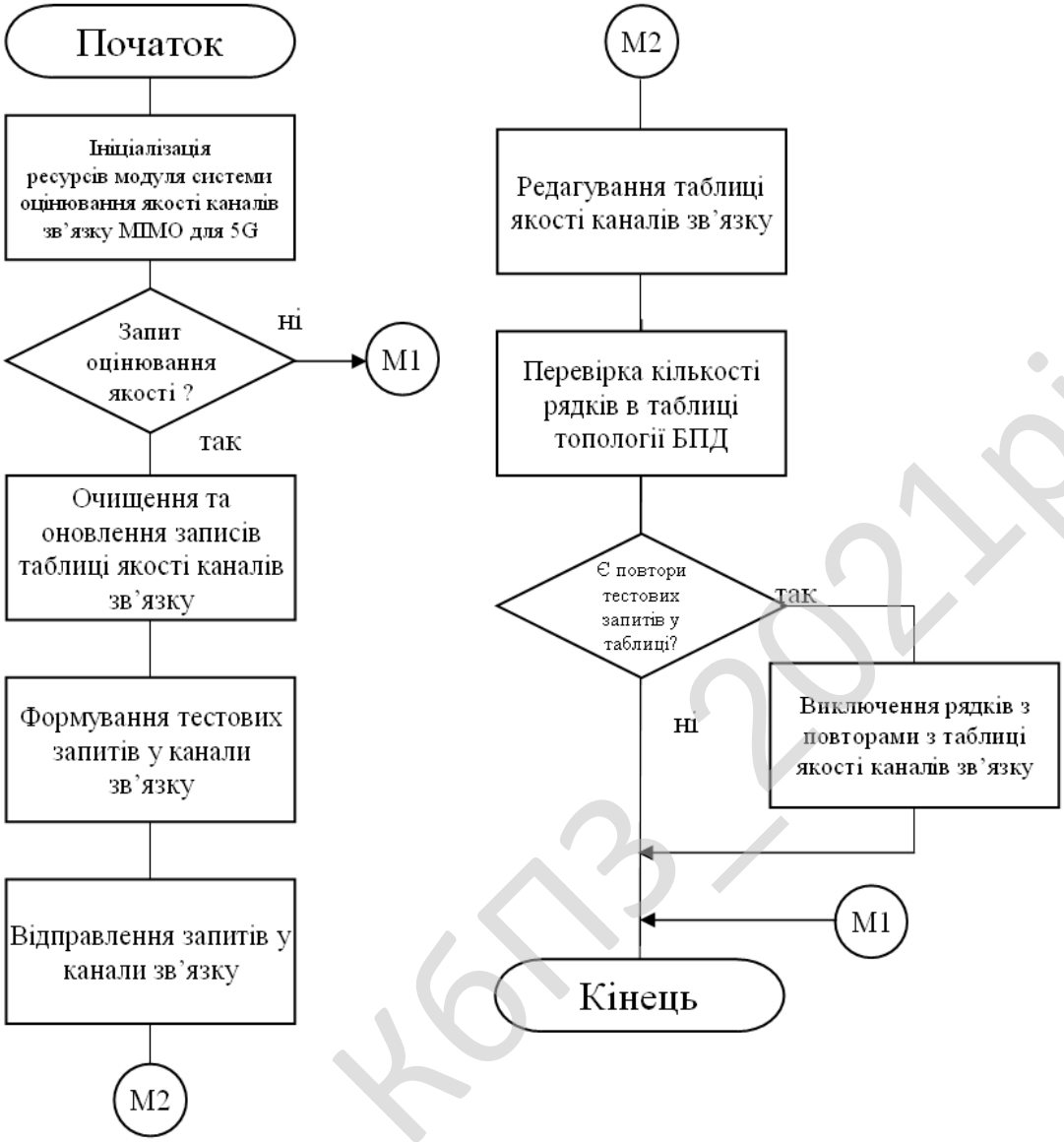

Рисунок 4.2 – Блок-схема роботи підпрограми

– Редагування таблиці якості каналів зв'язку.

– Перевірка кількості рядків в таблиці топології БПД.

– Є повтори тестових запитів у таблиці.

– Виключення рядків з повторами з таблиці якості каналів зв'язку.

Для реалізації системи оцінювання якості каналів зв'язку MIMO для 5G, а саме реалізації частини під задач необхідно розробити модуль який би дозволив отримати ім'я домену по його IP-адресі, дізнатися IP-адресу або доменне ім'я по його імені, дізнатися IP-адресу комп'ютера по його імені, дізнатися ім'я

комп'ютера по його IP-адресою, перевести IP-адреса комп'ютера в числовий формат, також провести підрахунок вхідного і вихідного трафіку через інтерфейси враховуючи весь трафік робочої станції. Для реалізації цих перетворень було створено модуль *DSp* розглянемо його вихідний код:

```
unit DSp; // назва модулю

interface // інтерфейс модулю

uses // стандартне підключення бібліотек модулю
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, Winsock, ExtCtrls, Math;

Type
// об’ява класу модулю – TForm7
  TForm7 = class(TForm)
    Button1: TButton;
    Label1: TLabel;
    Edit1: TEdit;
    Label2: TLabel;
    Button2: TButton;
    Bevel1: TBevel;
    Edit2: TEdit;
    Label3: TLabel;
    Label4: TLabel;
    Bevel2: TBevel;
    Button3: TButton;
    Edit3: TEdit;
    Label5: TLabel;
    Label6: TLabel;
    Bevel3: TBevel;
    Label7: TLabel;
    Bevel4: TBevel;
    Button4: TButton;
    Edit4: TEdit;
    Label8: TLabel;
    Label9: TLabel;
    Button5: TButton;
    Edit5: TEdit;
    Label10: TLabel;
    Label11: TLabel;
```

```
    Bevel5: TBevel;
    Timer1: TTimer;
    ListBox1: TListBox;
    Button6: TButton;
    Label12: TLabel;
    Label13: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Label4DblClick(Sender: TObject);
    procedure Label1DblClick(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Label6DblClick(Sender: TObject);
    procedure Label9DblClick(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Label11DblClick(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Button6Click(Sender: TObject);
  private
  public
  end;

var
// об'ява глобальних змінних модулю TForm7
  Form1: TForm7; // об'ява екземпляру класу
  stop_traf: boolean;
  count,trafbitin,trafbitout,trafbitold: integer;

implementation // реалізація

{$R *.dfm} // ресурс

function IPAddrToName(IPAddr: string): string;
var
 SockAddrIn: TSockAddrIn;
 HostEnt: PHostEnt;
 WSAData: TWSAData;
begin
 WSAStartup($101, WSAData);
 SockAddrIn.sin_addr.s_addr:=inet_addr(PChar(IPAddr));
 HostEnt:=GetHostByAddr(@SockAddrIn.sin_addr.S_addr, 4, AF_INET);
```

```
 if HostEnt<>nil
 then Result:=StrPas(Hostent^.h_name)
 else Result:='';
end;
// натискання 1 кнопки модулю TForm7
procedure TForm7.Button1Click(Sender: TObject);
begin
 Label1.Caption:='Name: '+IPAddrToName(Edit1.Text);
end;

// натискання 2 кнопки модулю TForm7
procedure TForm7.Button2Click(Sender: TObject);
var
 // Зберігаємо оригінальне значення IP адреси
// частини оригінального IP
 OrgVal: string;
O1,O2,O3,O4: string;
 // шістнадцяткові частини
 H1,H2,H3,H4: string;
 // Тут будуть зібрані всі шістнадцяткові частини
 HexIP: string;
 XN: array[1..8] of Extended;
 Flt1: Extended;
 Xc: Integer;
begin
 // Зберігаємо в зворотному порядку для простого випадку
Xn[8]:=IntPower(16,0);Xn[7]:=IntPower(16,1);
Xn[6]:=IntPower(16,2);Xn[5]:=IntPower(16,3);
Xn[4]:=IntPower(16,4);Xn[3]:=IntPower(16,5);
Xn[2]:=IntPower(16,6);Xn[1]:=IntPower(16,7);

// Зберігаємо оригінальний IP адресу
 OrgVal:=Edit2.Text;
 O1:=Copy(OrgVal,1,Pos('.',OrgVal)-1);Delete(OrgVal,1,Pos('.',OrgVal));
 O2:=Copy(OrgVal,1,Pos('.',OrgVal)-1);Delete(OrgVal,1,Pos('.',OrgVal));
 O3:=Copy(OrgVal,1,Pos('.',OrgVal)-1);Delete(OrgVal,1,Pos('.',OrgVal));
 O4:=OrgVal;
 H1:=IntToHex(StrToInt(O1),2);H2:=IntToHex(StrToInt(O2),2);
 H3:=IntToHex(StrToInt(O3),2);H4:=IntToHex(StrToInt(O4),2);

 // Отримуємо шестнадцатиричное значення IP адреси
 HexIP:=H1+H2+H3+H4;
```

```
 // Перетворимо це велике шестнадцатиричное значення в змінну Float
 Flt1:=0;
 for Xc:=1 to 8 do
  begin
    case HexIP[Xc] of
     '0'..'9': Flt1:=Flt1+(StrToInt(HexIP[XC])*Xn[Xc]);
     'A': Flt1:=Flt1+(10*Xn[Xc]);
     'B': Flt1:=Flt1+(11*Xn[Xc]);
     'C': Flt1:=Flt1+(12*Xn[Xc]);
     'D': Flt1:=Flt1+(13*Xn[Xc]);
     'E': Flt1:=Flt1+(14*Xn[Xc]);
     'F': Flt1:=Flt1+(15*Xn[Xc]);
    end;
  end;
 Label4.Caption:='Number: '+FloatToStr(Flt1);
end;
// однаразове натискання на 4 текст модулю TForm7
procedure TForm7.Label4DblClick(Sender: TObject);
begin
 Edit2.Text:=Label4.Caption;
end;
// двохразове натискання на 4 текст модулю TForm7
procedure TForm7.Label1DblClick(Sender: TObject);
begin
 Edit1.Text:=Label1.Caption;
end;

const // константа
  WINSOCK_VERSION=$0101;

// натискання 3 кнопки модулю TForm7
procedure TForm7.Button3Click(Sender: TObject);
var
 WSAData: TWSAData;
 p: PHostEnt;
begin
 WSAStartup(WINSOCK_VERSION, WSAData);
 p:=GetHostByName(PChar(Edit3.Text));
 Label6.Caption:='IP: '+inet_ntoa(PInAddr(p.h_addr_list^)^);
 WSACleanup;
end;
```

```
// повертає IP адреса
function LocalIP: string;
type
 TaPInAddr=array [0..10] of PInAddr;
 PaPInAddr=^TaPInAddr;
var
 phe:PHostEnt;
 pptr:PaPInAddr;
 Buffer:array [0..63] of char;
 i:Integer;
 GInitData:TWSADATA;
begin
 WSAStartup($101, GInitData);
 Result:='';
 GetHostName(Buffer, SizeOf(Buffer));
 phe:=GetHostByName(buffer);
 if phe=nil then Exit;
 pptr:=PaPInAddr(Phe^.h_addr_list);
 i:=0;
 while pptr^[i]<>nil do
  begin
   result:=StrPas(inet_ntoa(pptr^[i]^));
   Inc(i);
  end;
 WSACleanup;
end;

// дії при створенні форми
procedure TForm7.FormCreate(Sender: TObject);
begin
 Label7.Caption:='Local IP: '+LocalIP;
end;

// натискання 4 кнопки модулю TForm7
procedure TForm7.Button4Click(Sender: TObject);
var
 wsdata: TWSAData;
 hostName: array [0..255] of char;
 hostEnt: PHostEnt;
 addr: PChar;
begin
 WSAStartup ($0101, wsdata);
```

```
  try
   GetHostName(hostName, sizeof (hostName));
   StrPCopy(hostName,Edit4.Text);
   hostEnt:=GetHostByName(hostName);
   if Assigned(hostEnt)
   then
    if Assigned(hostEnt^.h_addr_list)
    then
     begin
      addr:=hostEnt^.h_addr_list^;
      if Assigned(addr)
      then
       begin
        Label9.Caption:=Format('%d.%d.%d.%d',[byte(addr[0]),
        byte(addr[1]),byte(addr[2]),byte(addr[3])]);
       end;
     end;
  finally
   WSACleanup;
  end;
end;

procedure TForm7.Label6DblClick(Sender: TObject);
begin
 Edit3.Text:=Label6.Caption;
end;

procedure TForm7.Label9DblClick(Sender: TObject);
begin
 Edit4.Text:=Label9.Caption;
end;
// порівняння
function IPAddrToCompName(IPAddr: string): string;
var
 SockAddrIn: TSockAddrIn;
 HostEnt: PHostEnt;
 WSAData: TWSAData;
begin
 WSAStartup($101, WSAData);
 SockAddrIn.sin_addr.s_addr:=inet_addr(PChar(IPAddr));
 HostEnt:=gethostbyaddr(@SockAddrIn.sin_addr.S_addr, 4, AF_INET);
 if HostEnt<>nil
```

```
 then Result:=StrPas(Hostent^.h_name)
 else Result:='';
end;

// натискання 5 кнопки модулю TForm7
procedure TForm7.Button5Click(Sender: TObject);
begin
 Label11.Caption:='Name: '+IPAddrToCompName(Edit5.Text);
end;

procedure TForm7.Label11DblClick(Sender: TObject);
begin
 Edit5.Text:=Label11.Caption;
end;

// трафік
type
 TMibIfRow = packed record
  wszName           : array[0..255] of WideChar;
  dwIndex           : DWORD;
  dwType            : DWORD;
  dwMtu             : DWORD;
  dwSpeed           : DWORD;
// визначає поточну швидкість передавання даних в секунду
  dwPhysAddrLen     : DWORD;
  bPhysAddr         : array[0..7] of Byte;
// содержит физический адрес интерфейса
  dwAdminStatus     : DWORD;
  dwOperStatus      : DWORD;
  dwLastChange      : DWORD;
  dwInOctets        : DWORD;
// містить кількість байт прийнятих через інтерфейс
  dwInUcastPkts     : DWORD;
  dwInNUCastPkts    : DWORD;
  dwInDiscards      : DWORD;
  dwInErrors        : DWORD;
  dwInUnknownProtos: DWORD;
  dwOutOctets       : DWORD;
// містить кількість байт відправлених інтерфейсом
  dwOutUCastPkts    : DWORD;
  dwOutNUCastPkts   : DWORD;
  dwOutDiscards     : DWORD;
```

```
  dwOutErrors      : DWORD;
  dwOutQLen        : DWORD;
  dwDescrLen       : DWORD;
  bDescr           : array[0..255] of Char;
// містить опис інтерфейсу
 end;
 TMibIfArray = array [0..512] of TMibIfRow;
 PMibIfRow = ^TMibIfRow;
 PMibIfArray = ^TMibIfArray;

type
 TMibIfTable = packed record
   dwNumEntries: DWORD;
   Table       : TMibIfArray;
 end;
 PMibIfTable = ^TMibIfTable;

var
 GetIfTable:function(pIfTable: PMibIfTable; pdwSize: PULONG;
                              bOrder: Boolean): DWORD; stdcall;

// Інтерфейси

function WSAIoctl(s: TSocket; cmd: DWORD; lpInBuffer: PCHAR; dwInBufferLen:
 DWORD;
 lpOutBuffer: PCHAR; dwOutBufferLen: DWORD;
 lpdwOutBytesReturned: LPDWORD;
 lpOverLapped: POINTER;
 lpOverLappedRoutine: POINTER): integer; stdcall; external 'WS2_32.DLL';

const
 SIO_GET_INTERFACE_LIST = $4004747F;
 IFF_UP = $00000001;
 IFF_BROADCAST = $00000002;
 IFF_LOOPBACK = $00000004;
 IFF_POINTTOPOINT = $00000008;
 IFF_MULTICAST = $00000010;

type
 sockaddr_gen = packed record
  AddressIn: sockaddr_in;
  filler: packed array [0..7] of char;
```

```
end;

type
 INTERFACE_INFO = packed record
  iiFlags: u_long;
// прапори інтерфейсу
  iiAddress: sockaddr_gen;
// Адреса інтерфейсу
  iiBroadcastAddress: sockaddr_gen;
// Broadcast адреса
  iiNetmask: sockaddr_gen;
// Маска підмережі
end;

function EnumInterfaces(var sInt: string): Boolean;
var
 s: TSocket;
 wsaD: WSADATA;
 NumInterfaces: Integer;
 BytesReturned: u_long;
 pAddrInet: SOCKADDR_IN;
 pAddrString: PChar;
 PtrA: pointer;
 Buffer: array[0..20] of INTERFACE_INFO;
 i: integer;
begin
 result:=true;
// инициализируем змінну
 sInt:='';
 WSAStartup($0101, wsaD);
// запускаємо WinSock
// тут можна дабавіть різні обробники помилки
  s: = Socket (AF_INET, SOCK_STREAM, 0);
 // відкриваємо сокет
 if (s=INVALID_SOCKET)
 then Exit;
 try
// викликаемо WSAIoCtl
  PtrA:=@bytesReturned;
  if (WSAIoCtl(s, SIO_GET_INTERFACE_LIST, nil, 0, @Buffer,
                          1024, PtrA, nil, nil)<>SOCKET_ERROR)
  then
```

```
   begin
// якщо OK, то визначаємо кількість існуючих інтерфейсів
     NumInterfaces:=BytesReturned div SizeOf(INTERFACE_INFO);
     for i:=0 to NumInterfaces-1 do
// для кожного інтерфейсу
      begin
       pAddrInet:=Buffer[i].iiAddress.AddressIn;
// IP адреса
       pAddrString:=inet_ntoa(pAddrInet.sin_addr);
       if pAddrString<>'127.0.0.1'
       then
        begin
         sInt:=sInt+'IP = '+pAddrString+', '+#10#13;
         pAddrInet:=Buffer[i].iiNetMask.AddressIn;
// маска підмережі
         pAddrString:=inet_ntoa(pAddrInet.sin_addr);
         sInt:=sInt+' Mask='+pAddrString+',';
        end
       else sInt:='IP = "localhost"';
      end;
  end;
 except
end;

 // закриваємо сокети
 CloseSocket(s);
 WSACleanUp;
 result:=false;
end;
// перетворення типів Bytes у String
function BytesToString(Value: integer): string;
const
 OneKB=1024;
 OneMB=OneKB*1024;
 OneGB=OneMB*1024;
begin
 if Value<OneKB
 then Result:=FormatFloat('#,##0.00 B',Value)
 else
  if Value<OneMB
  then Result:=FormatFloat('#,##0.00 KB', Value/OneKB)
  else
```

```
   if Value<OneGB
   then Result:=FormatFloat('#,##0.00 MB', Value/OneMB)
end;

procedure TForm7.Timer1Timer(Sender: TObject);
// допоміжна функція, яка перетворює MAC адресу до
  // "нормального" виду визначаємо спеціальний тип, щоб
  // можна було передати в функцію массив
type
 TMAC=array [0..7] of Byte;
// в якості першого значення масив, друге значення,
  // розмір даних в масиві
function GetMAC(Value: TMAC; Length: DWORD): string;
var
 i: integer;
begin
 if Length=0
 then Result:='00-00-00-00-00-00'
 else
  begin
   Result:='';
   for i:=0 to Length-2 do
    Result:=Result+IntToHex(Value[i],2)+'-';
    Result:=Result+IntToHex(Value[Length-1],2);
   end;
end;

var
 FLibHandle: THandle;
 Table: TMibIfTable;
 i, Size: integer;
 s,trafnormin,trafnormout: string;
begin
 Timer1.Enabled:=false;
 // припиняємо про всяк випадок таймер
 ListBox1.Items.BeginUpdate;
 ListBox1.Items.Clear;
// очищаємо список
 FLibHandle:=LoadLibrary('IPHLPAPI.DLL');
// завантажуємо бібліотеку
 if FLibHandle=0
 then Exit;
```

```
 @GetIfTable:=GetProcAddress(FLibHandle, 'GetIfTable');
 if not Assigned(GetIfTable)
 then
  begin
   FreeLibrary(FLibHandle);
   Close;
  end;
 //
 Size:=SizeOf(Table);
 if GetIfTable(@Table,@Size,false)=0
 then
// виконуємо функцію
  for i:=0 to Table.dwNumEntries-1 do
// кількість мережевих карт
   begin
    with ListBox1.Items do
     begin
// виводимо результати
    if string(GetMAC(TMAC(Table.Table[i].bPhysAddr),
              Table.Table[i].dwPhysAddrLen))<>'00-00-00-00-00-00'
// порівняння MAC адрес
      then
       begin
Add('Опис: '+string(Table.Table[i].bDescr));
// найменування інтерфейсу
Add('MAC-adress: '+string(GetMAC(TMAC(Table.Table[i].bPhysAddr),
    Table.Table[i].dwPhysAddrLen)));
// MAC адреса
// переклад до нормальних одиницям "Вхідного" трафіку
trafbitin:=Table.Table[i].dwInOctets;
// всього прийнято байт
         trafnormin: = BytesToString (trafbitin);
// переклад до нормальних одиницям "Вихідного" трафіку
          trafbitout: = Table.Table [i] .dwOutOctets;
// всього відправлено байт
          trafnormout: = BytesToString (trafbitout);
// скидання трафіку
        if stop_traf=true
        then
         begin
          trafbitold:=trafbitin;
          trafnormin:='0,00 B';
```

```
          trafnormout:='0,00 B';
         end;
// новий трафік більше старого
        if trafbitin>=trafbitold
        then
         begin
          trafbitin:=trafbitin-trafbitold;
          trafnormin:=BytesToString(trafbitin);
         end
        else
// новий трафік менше старого
         begin
          trafbitin:=trafbitold;
          trafnormin:=BytesToString(trafbitin);
         end;
        Add(' всього прийнято байт: '+trafnormin);
        Add(' всього відправлено байт: '+trafnormout);       end;
     end;
   end;
EnumInterfaces(s);
 ListBox1.Items.Add(s);
ListBox1.Items.EndUpdate;
 FreeLibrary(FLibHandle);
// активувати таймер
 Timer1.Enabled:=true;
end;
procedure TForm7.Button6Click(Sender: TObject);
begin
 if stop_traf=false then stop_traf:=true
 else stop_traf:=false;
end;
end.
```

Redmine – вільне серверне ПЗ для управління проектами та відстежування помилок. До системи входить календар-планувальник та діаграми Ганта для візуального представлення ходу робіт за проектом та строків виконання. Redmine написано на мові Ruby і є ПЗ розробленим з використанням відомого веб-фреймворку Ruby on Rails, що означає легкість в розгортанні системи та її адаптації під конкретні вимоги. Для кожного проекту можна вести свої вікі та форуми.

Функціональні можливості:

– Ведення декількох проектів.

– Гнучка система доступу з використанням ролей.

– Система відстеження помилок.

– Діаграми Ганта та календар.

– Ведення новин проекту, документів та управління файлами.

– Сповіщення про зміни за допомогою RSS-потоків та електронної пошти.

– Власна Wiki для кожного проекту.

– Форуми для кожного проекту.

– Облік часових витрат.

– Налаштування власних (custom) полів для задач, затрат часу, проектів та користувачів.

– Легка інтеграція із системами керування версіями (SVN, CVS, Git, Mercurial, Bazaar и Darcs).

– Створення записів про помилки на основі отриманих листів

– Підтримка LDAP автентифікації.

– Можливість самореєстрації нових користувачів.

– Багатомовний інтерфейс (у тому числі українська мова).

– Підтримка СКБД: MySQL, PostgreSQL, SQLite.

Діаграма Ганта (*Gantt chart*, також стрічкова діаграма, графік Ганта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка робіт за будь-яким проектом. Є одним з методів планування та управління проектами.

Діаграма Ганта являє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями.

Діаграма може використовуватися для представлення поточного стану виконання робіт: частина прямокутника, що відповідає завданню, заштриховується, відзначаючи відсоток виконання завдання; показується вертикальна лінія, що відповідає моменту «сьогодні».

Часто діаграма Ганта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, зображеній на діаграмі, а стовпці містять додаткову інформацію про задачу.

Система відстеження помилок Багтрекер – прикладна програма для допомоги розробникам програмного забезпечення (програмістам, тестувальникам тощо) враховувати і контролювати помилки, знайдені у програмах, питання щодо функціональності, рішення та оновлення, побажання користувачів, а також стежити за процесом їх виконання.

Кожному, хто розробляв програмні продукти, добре знайоме співвідношення «20/80» – останні 20 % роботи тривають 80 % часу.

Як це не парадоксально, але нічого дивного в цій пропорції немає, адже саме на завершальній стадії починається тестування проекту, коли виявляються помилки, і що більший проект, то більше буде знайдено помилок.

Водночас досить часто виявляється, що більшість цих помилок були відомі та могли бути виправлені з меншими витратами на попередніх стадіях роботи, але не були вчасно описані, а потім загубилися серед інших важливих завдань.

Отже, система відстеження помилок у найпростішому варіанті – це процес, що включає в себе виявлення помилки, її опис, виправлення і перевірку цього виправлення, тобто процес «стеження» за багом протягом всього як його життєвого циклу, так і життєвого циклу розробки в цілому.

Сукупність інформації про дефект. Головний компонент такої системи – база даних, що містить відомості про виявлені дефекти. Ці відомості можуть включати в себе:

– номер (ідентифікатор) дефекту;

– хто повідомив про дефект;

– дата і час виявлення дефекту;

– версія продукту, в якій виявлено дефект;

– серйозність (критичність) дефекту та пріоритет рішення;

– опис кроків для відтворення дефекту (неправильної поведінки програми);

– відповідальний за усунення дефекту;

– обговорення можливих рішень та їх наслідків;

– поточний стан виправлення дефекту;

– версії продукту, в якій дефект виправлений.

Крім того, розвинені системи надають можливість прикріплювати файли, які допомагають описати проблему, наприклад, дамп пам'яті або скріншот.

Використання. Основна перевага систем відстеження помилок полягає в забезпеченні чітких централізованих оглядів, запитів на розробку (включаючи помилки і виправлення) та їх стан. У корпоративному середовищі, системи відстеження помилок можуть бути використані для генерації звітів по продуктивності програмістів виправлення помилок. Однак, це може іноді приводити до неточних результатів, тому що різні помилки можуть мати різні ступені пріоритету та серйозності, що пов'язано з складністю їх фіксації.

Життєвий цикл дефекту. Як правило, система відстеження помилок використовує той чи інший варіант «життєвого циклу» помилки, стадія якого визначається поточним станом помилки.

Типовий життєвий цикл дефекту:

1. Новий – дефект зареєстрований тестувальником.

2. Призначений – призначений відповідальний за виправлення дефекту.

3. Дозволений – дефект переходить назад у сферу відповідальності тестувальника. Як правило, супроводжується резолюцією, наприклад:

– Виправлено (виправлення включені у версію таку-то).

– Дубль (повторює дефект, що вже знаходиться в роботі).

– Не виправлено (працює відповідно до специфікації, має занадто низький пріоритет, виправлення відкладено до наступної версії тощо).

– «В мене все працює» (запит додаткової інформації про умови, в яких дефект проявляється).

4. Далі тестувальник проводить перевірку виправлення, залежно від чого дефект або знову переходить у стан «Призначений» (якщо він описаний як виправлений, але не виправлений), або у стан «Закрито».

5. Відкрито повторно – дефект знайдено знову в іншій версії.

Система може надавати адміністраторові можливість налаштування користувачі, які можуть переглядати і редагувати помилки залежно від їх стану, переводити їх в інший стан або видаляти.

У корпоративному середовищі, система відстеження помилок може використовуватися для отримання звітів, що показують продуктивність програмістів при виправленні помилок. Однак, часто такий підхід не дає достатньо точних результатів через те, що різні помилки мають різну ступінь серйозності та складності. При цьому серйозність проблеми прямо не стосується складності її усунення.

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.

Було використано розши́рювану мо́ву розмі́тки (Extensible Markup Language, скорочено XML) – запропоновану консорціумом World Wide Web Consortium (W3C) стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет. Є спрощеною підмножиною мови розмітки SGML.

XML-документ складається із текстових знаків, і придатний до читання людиною. Стандарт XML (Recommendation, перше видання від 10 лютого 1998, останнє, четверте видання 29 вересня 2006) визначає набір базових лексичних та

синтаксичних правил для побудови мови описання інформації шляхом застосування простих тегів.

Цей формат достатньо гнучкий для того, аби бути придатним для застосування в різних галузях. Іншими словами, запропонований стандарт визначає метамову, на основі якої шляхом запровадження обмежень на структуру та зміст документів визначаються специфічні, предметно-орієнтовані мови розмітки даних. Ці обмеження описуються мовами схем (Schema), такими як XML Schema (XSD), DTD або RELAX NG. Прикладами мов, заснованих на XML, є: XSLT, XAML, XUL, RSS, MathML, GraphML, XHTML, SVG, а також XML Schema.

Основні поняття. Коректність.

Коректний документ (well-formed document) відповідає всім синтаксичним правилам XML. Документ, що не є коректним, не може називатись XML-документом. Сумісний синтаксичний аналізатор (Conforming parser) не повинен обробляти такі документи. Зокрема, коректний XML-документ має:

1. Лише один елемент у корені.

2. Непорожні елементи розмічено початковим та кінцевим тегами (наприклад, <пункт> Пункт 1</пункт>). Порожні елементи можуть позначатися «закритим» тегом, наприклад <IAmEmpty />. Така пара еквівалентна <IAmEmpty></IAmEmpty>.

3. Один елемент не може мати декілька атрибутів з однаковою назвою. Значення атрибутів перебувають або в одинарних ('), або у подвійних (") лапках.

4. Теги можуть бути вкладені, але не можуть перекриватись. Кожен некореневий елемент мусить повністю перебувати в іншому елементі.

5. Документ має складатися тільки з правильно закодованих дозволених символів Юнікоду. Єдиними кодуваннями, які обов'язково має розуміти XML-процесор, є UTF-16 та UTF-8. Фактичне та задекларованe кодування (character encoding) документа мають збігатись. Кодування може бути задекларовано ззовні, як у заголовку «Content-Type» при передачі по протоколу HTTP, або в самому

документі використанням явної розмітки на самому початку документа. У разі відсутності інформації про кодування, документ має бути в кодуванні UTF-8 (або його підмножині ASCII).

Валідність

Документ називається валідним (valid), якщо він є коректним, містить посилання на граматичні правила та повністю відповідає обмеженням, вказаним у цих правилах (DTD або XML Schema або іншому подібному документі).

Синтаксичний аналізатор

Синтаксичним аналізатором (часто парсер, від parser) називається програма або компонент, що читає XML-документ, проводить синтаксичний аналіз та відтворює його структуру. Якщо синтаксичний аналізатор перевіряє документ на валідність, то такий аналізатор називають валідатором (validating).

Назви елементів чутливі до регістру літер. Наприклад, наступна пара елементів правильна: <Step> … </Step> у той час як ця – ні: <Step> … </step>.

Правильний вибір назв для XML-елементів підкреслюватиме значення даних у створеній мові розмітки. Це сприятиме полегшенню роботи людей з такими документами, зберігаючи можливості для комп'ютерної обробки даних.

Вибір змістовних назв передає семантику елементів та атрибутів для людини, без посилання на зовнішню документацію. Однак це може призвести до надмірності розмітки, що ускладнює редагування й збільшує розмір файлів.

Правильний вибір назв для XML-елементів підкреслюватиме значення даних у створеній мові розмітки. Це сприятиме полегшенню роботи людей з такими документами, зберігаючи можливості для комп'ютерної обробки даних. Вибір змістовних назв передає семантику елементів та атрибутів для людини, без посилання на зовнішню документацію. Однак це може призвести до надмірності розмітки, що ускладнює редагування й збільшує розмір файлів. XML-документи мають як фізичну, так і логічну структуру.

Коректність XML-документів

Залишивши назви, дозволену ієрархію, та значення елементів і атрибутів відкритою та можливою бути визначеною в спеціалізованих схемах або визначеннях типу документа (DTD).

XML утворює синтаксичну основу для створення спеціалізованих, заснованих на XML мовах розмітки даних.

Загальний синтаксис таких документів стабільний і наперед визначений – документи мають відповідати базовим вимогам XML, гарантуючи те, що довільне програмне забезпечення з підтримкою XML буде здатне щонайменше зчитати і відтворити відносну структуру інформації, що міститься в них.

Схема лише доповнює синтаксичні правила множиною обмежень. Зазвичай схеми обмежують назви елементів та атрибутів, дозволені типи значень і допустиму ієрархію елементів, наприклад, дозволяючи лише елементу з назвою «народження» містити під-елемент з назвою «місяць» та з назвою «день», і кожен із них мусить містити лише літери. Обмеження, вказані в схемі, можуть також включати присвоєння певних типів даних для впливу на те, як обробляється інформація; наприклад, дані елемента «місяць» можна визначити як такі, що містять лише місяць, як це визначено відповідно до використаної мови схем.

Коректний XML-документ, що відповідає обмеженням схеми або DTD, називається валідним.

DTD (Document Type Definition). Найдавнішим форматом схем для XML є успадкований від SGML формат визначення типу документа (Document Type Definition, DTD). У той час, як через включення до стандарту XML 1.0 DTD став поширеним форматом схем, він має такі обмеження:

1. Відсутність нових можливостей XML, із найважливішою з посеред них простори назв.

2. Брак виразності. Деякі формальні аспекти XML-документів неможливо відобразити в DTD.

3. Використовується спеціалізований, заснований не на XML синтаксис для опису схем.

DTD все ще використовується в багатьох програмах, оскільки він вважається найпростішим форматом для аналізу та збереження.

XML Schema. На заміну DTD було розроблено нову мову схем – XML Schema (буквально – XML-схема), скорочено позначається як XSD (від XML Schema Definition). XSD набагато потужніші за DTD в описанні заснованих на XML мов. Вони використовують багатий набір типів даних, підтримують детальніші обмеження на структуру документів, та повинні оброблятись складнішими системами. XSD побудовано на основі XML, що робить можливим використання звичайних інструментів XML для їхньої обробки, хоча реалізації XSD вимагають набагато більше аніж просто можливість читати XML.

Серед недоліків XSD називають такі:

1.Специфікація дуже велика, що робить її складною для розуміння та реалізації.

2. Оснований на XML синтаксис додає надмірності мові, що ускладнює читання та запис XSD.

3. Валідація відносно схеми може бути дорогим додатком до синтаксичного аналізу XML-документів.

4. Можливості моделювання дуже обмежені, без можливості впливу значень атрибутів на вміст елементів.

5. Модель отримання типів даних є дуже обмеженою, зокрема в тому, що отримання шляхом розширення є рідко коли корисним.

6. Механізми ключа/посилання на ключ/унікальності не враховують тип даних.

7. Концепція PSVI (Post Schema Validation Infoset) не має стандартного представлення або прикладного програмного інтерфейса, що працює проти незалежності від реалізації, якщо не виконується повторна валідація.

RELAX NG є іншою поширеною мовою схем для XML. Вперше RELAX

NG було визначено стандартом OASIS, а тепер – міжнародним стандартом ISO (як частина DSDL). Ця мова схем має два формати: оснований на XML, та компактний, не-XML. Компактний синтаксис призначений для покращення можливості читання та написання схем, однак, оскільки існує точно визначений спосіб перетворення компактного формату в оснований на XML, і навпаки, не втрачаються переваги від використання стандартних XML-інструментів. RELAX NG має простіші системи для визначення та валідації у порівнянні з XML Schema, що робить її привабливішою для використання та реалізації. Також існує можливість використання модулів роботи з типом даних; наприклад, автор схеми RELAX NG може вказати, що значення XML-документа мають відповідати визначенням типам даних у форматі XML Schema Datatypes.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму DES.

DES є класичною мережею Фейштеля із двома гілками. Дані шифруються 64-бітними блоками, використовуючи 56-бітний ключ. Алгоритм перетворить за кілька раундів 64-бітний вхід в 64-бітний вихід. Довжина ключа дорівнює 56 бітам. Процес шифрування складається із чотирьох етапів. На першому з них виконується початкова перестановка (IP) 64-бітного вихідного тексту (забілювання), під час якої біти переставляються у відповідності зі стандартною таблицею.

Наступний етап складається з 16 раундів однієї й тої ж функції, що використовує операції зрушення й підстановки. На третьому етапі ліва й права половини виходу останньої (16-й) ітерації міняються місцями.

Нарешті, на четвертому етапі виконується перестановка IP^{-1} результату, отриманого на третьому етапі. Перестановка IP^{-1} інверсна початковій перестановці.

Рисунок 4.3 – Загальна схема DES

Праворуч на рисунку показаний спосіб, яким використовується 56-бітний ключ. Спочатку ключ подається на вхід функції перестановки.

Потім для кожного з 16 раундів підключ K_i є комбінацією лівого циклічного зрушення й перестановки.

Функція перестановки та сама для кожного раунду, але підключи K_i для кожного раунду виходять різні внаслідок повторюваного зрушення біт ключа.

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: Виведення вікна проектування архітектури; Обрання набору символів; Локальне сканування; Вікна поточної архітектури; Налаштування.

– Верхнього меню: Файл; Статистика сканування; Довідка.

– Розділу виведення результату роботи системи у графічному вигляді.

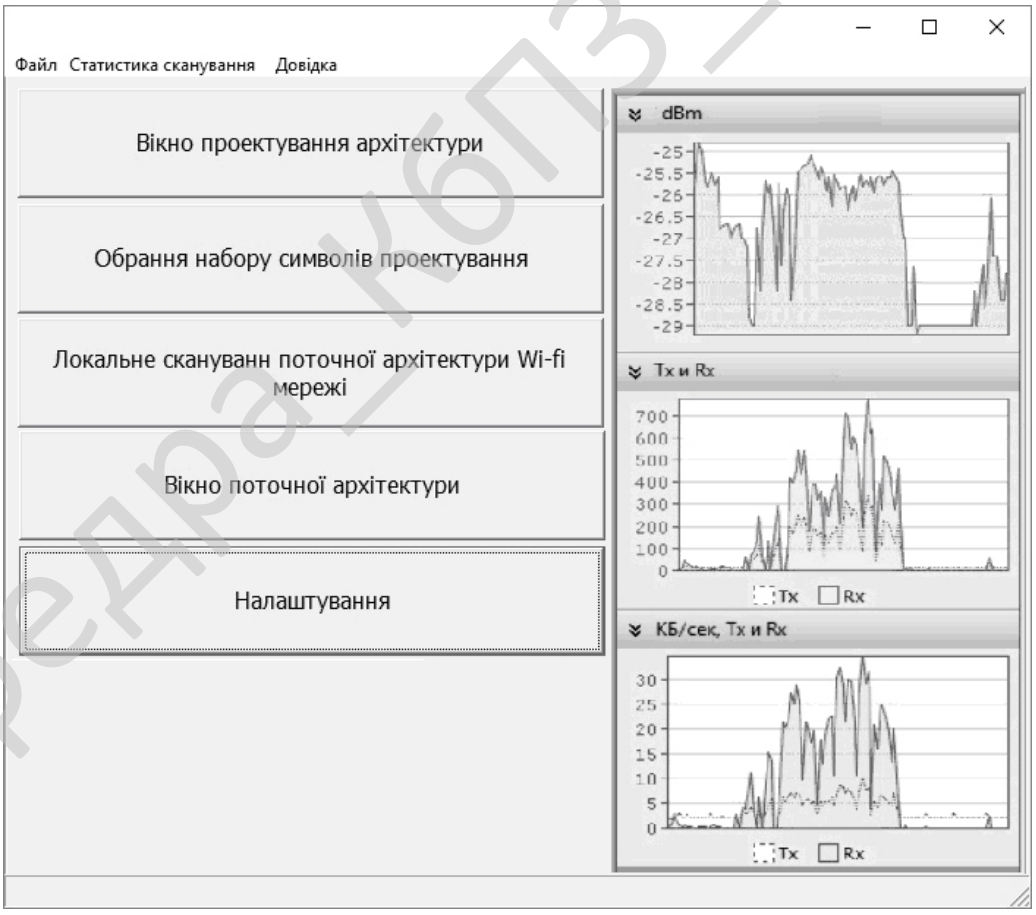

Рисунок 5.1 – Головне вікно ПЗ

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.

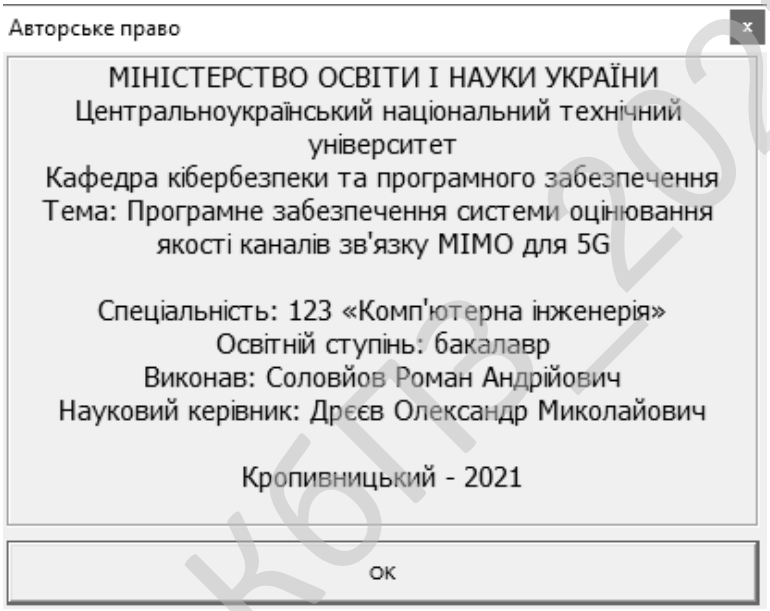

Рисунок 5.2 – Авторське право

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 оцінювання якості каналів зв'язку MIMO для 5G.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем оцінювання якості каналів зв'язку MIMO для 5G.

– Досліджена система оцінювання якості каналів зв'язку MIMO для 5G.

– На основі отриманих результатів досліджень створена програмна реалізація системи оцінювання якості каналів зв'язку MIMO для 5G.

Розроблені під час виконання кваліфікаційної бакалаврської роботи алгоритми дозволяють успішно вирішувати завдання оцінювання якості каналів зв'язку MIMO для 5G.

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 Delphi 10.4 Sydney. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи оцінювання якості каналів зв'язку MIMO для 5G. 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 DES.

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. Osseiran A. et al. Scenarios for 5G mobile and wireless communications: the vision of the METIS project // IEEE Communications Magazine. 2014. V. 52. № 5. P. 26-35.

2. Hussain Sk.S. et al. An overview of massive MIMO system in 5G // International Science Press, I J C T A. 2016. P. 4957-4968.

3. Study on Scenarios and Requirements for Next Generation Access Technologies // 3GP TR 38.913 V. 0.3. 2016. P. 36.

4. Larsson E.G. et al. Massive MIMO for next generation wireless systems //IEEE Communications Magazine. 2014. V. 52. № 2. P. 186-195.

5. Marzetta T.L. Noncooperative cellular wireless with unlimited numbers of base station antennas // IEEE Transactions on Wireless Communications. 2010. V. 9. № 11. P. 3590-3600.

6. Rusek F. et al. Scaling up MIMO: Opportunities and challenges with very large arrays // IEEE Signal Processing Magazine. 2013. V. 30. № 1. P. 40-60.

7. Larsson E.G. et al. Massive MIMO for next generation wireless systems // IEEE Communications Magazine. 2014. V. 52. № 2. P. 186-195.

8. Ngo H.Q., Larsson E.G., Marzetta T.L. Energy and spectral efficiency of very large multiuser MIMO systems // IEEE Transactions on Communications. 2013. V. 61. № 4. P. 1436-1449.

9. Gao X. et al. Massive MIMO performance evaluation based on measured propagation data // IEEE Transactions on Wireless Communications. – 2015. V. 14. № 7. P. 3899-3911.

10. Одоевский С., Степанец В. Планировать беспроводную связь с комфортом: программный комплекс ONEPLAN RPLS (ONEGA) // ПЕРВАЯ МИЛЯ. 2013. № 2. С. 34-39.

11. Фокин Г.А. Управление самоорганизующимися пакетными радиосетями на основе радиостанций с направленными антеннами: автореф. дисс. на соискание науч. степени канд. техн. наук: спец. 05.13.13 "Телекоммуникационные системы и компьютерные сети". – СПб, 2009.

12. Mitsubishi Electric's New Multibeam Multiplexing 5G Technology Achieves 20 Gbps Throughput // Mitsubishi Electrics, http://www.mitsubishielectric.com/news/2016/0121.html. – 21 янв. 2016.

13. Views on Massive MIMO for New Radio // 3GPP Nanjing, Written Contributions R1-165063. – 2016.

14. Lu L. et al. An overview of massive MIMO: Benefits and challenges // IEEE journal of selected topics in signal processing. 2014. V. 8. № 5. P. 742-758.

15. Зюко А.Г. и др. Теория передачи сигналов. – М.: Радио и связь, 1980.

16. Одоевский С., Степанец В., Зибарев Е., Болкунов А., Зайченко А. Беспроводная связь и принцип "не навреди": ПК ONEPLAN Sazon // ПЕРВАЯ МИЛЯ. 2017. №4. С. 52-57.

17. Harris P., Malkowsky S. Setting a World Record in 5G Wireless Spectrum Efficiency With Massive MIMO // Virtuelle Instrumente in der Praxis. 2016. V. 21. P. 272-277.

18. Zhang X., Zhou X. LTE-advanced air interface technology. – Boca Raton: CRC Press, 2012.

19. Hochwald B.M., Marzetta T.L., Tarokh V. Multiple-antenna channel hardening and its implications for rate feedback and scheduling // IEEE transactions on Information Theory. 2004. V. 50. № 9. P. 1893-1909.

20. 20.Bjцrnson E. Channel hardening makes fading channels behave as deterministic // https://ma-mimo.ellintech.se/2017/01/25/channel-hardening-makes-fading-channels-behave-as-deterministic/. – 25 янв. 2017.

21. Study on New Radio Access Technology Physical Layer Aspects // 3GPP TR 38.802 V.14.2.0–2017.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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 ІТ-індустрії: зб. тез VII міжнар. наук.-практ. конф., м. Харків, 17-18 квітня 2015 р. – Х.: ХНЕУ, 2015. – С. 14.

41. 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.

42. 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.

43. 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.

44. 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.

45. Смирнов С. А. Метод управления доступом к облачным телекоммуникационным 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.

46. 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.

47. 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.

48. 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.

49. 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.

50. 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.

51. 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.

52. 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.

53. 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.

54. 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.

55. 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.

56. 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.

57. 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.

58. 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 ІІІ міжнар. наук.-практ. конф., м. Харків, 28-30 кві. 2016 р. – Х.: ХННІ ДВНЗ «УБС», 2016. – С. 178-182.

59. 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.

60. 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.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи оцінювання якості каналів зв’язку MIMO для 5G.

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 оцінювання якості каналів зв’язку MIMO для 5G.

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 оцінювання якості каналів зв’язку MIMO для 5G;

– 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 забезпечення.

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 Delphi 10.4 Sydney.

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 – 90 аркушів.

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 22.05.2020 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист 5.06.2020 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 оцінювання якості каналів зв’язку MIMO для 5G

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 NGN
  N:=TNetworkNeighborhood.Create;
  try
   // Отримання списку всіх комп' ютерів в мережі NGN
   Memo1.Lines.Clear;
   N.ListComputers(Memo1.Lines);

   // Отримання списку всіх робочих груп і комп' ютерів в мережі NGN,
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 NGN
   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 каналів зв'язку MIMO для 5G

```
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 NGN }

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 Cisco}
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 Cisco 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 NGN 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 Cisco : '  + 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( ‘ Exo запит        : ‘  + IntToStr( dwEchos ) );
      ICMPIn.Add( ‘ Exo 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( ‘Exo запит        : ‘  + IntToStr( dwEchos ) );
      ICMPOut.Add( ‘Exo 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 NGN знайдені ідентичні імена' ;
  SFileError053 = '  - мережний шлях не знайдено' ;
  SFileError054 = '  - мережа зайнята' ;
  SFileError055 = '  - ресурс мережі NGN або пристрій недоступний' ;
  SFileError057 = '  - апаратна помилка в мережному адаптері' ;
  SFileError058 = '  - сервер не в змозі виконувати дану дію' ;
  SFileError059 = '  - помилка у мережі NGN' ;
  SFileError064 = '  - недоступне мережне ім"я' ;
  SFileError065 = '  - немає доступу до мережі NGN' ;
  SFileError066 = '  - невірно вказатий тип мережного ресурсу' ;
  SFileError067 = '  - не знайдене вказане мережне ім"я' ;
  SFileError070 = '  - відключейний сервер мережі NGN' ;
  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 мережею каналів зв'язку MIMO для 5G

```
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
NGN}
  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 NGN }

    procedure Refresh;

    { містить списки всіх комп' ютерів в  мережі NGN}

    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 NGN немає }

    function FindComputer(Name: TString): TString;

    { Процедура ListComputers копіює список всіх комп' ютерів в мережі NGN
      у об' єкті TStrings}
    procedure ListComputers(Strings: TStrings);

    { Процедура ListNetwork копіює відсортований в алфавітному порядку список
всіх робочих груп і комп' ютерів в мережі NGN.
     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 NGN }

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



{ Функція EnumSharedResources перераховує загальні ресурси мережі NGN. 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.
```